Premios para los lectores • Estrategia • C y Ensamblador • Aventura • DIV 2 • Arcade

La primera revista dedicada a aprender a programar videojuegos

DIV manía

www.prensatecnica.com

Año 1 • Número 1



ESPECIAL TOKENKAI

Todo lo que debes saber sobre este próximo título de estrategia, el primer juego profesional programado con DIV

DIBUJO Y DISEÑO

Todos los trucos necesarios para que aprendas a moverte entre imágenes

CONCURSO JUEGOS DIV

INICIACIÓN DIV

Si aún no has tenido ocasión de acercarte a el, aprende a utilizar este entorno desde el principio

PROGRAMA TUS JUEGOS

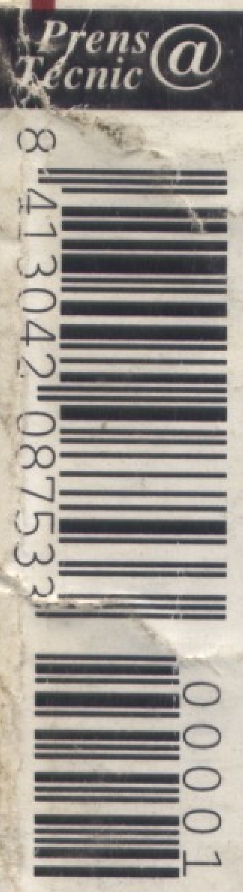
Rol, estrategia, aventuras... Los secretos de cada género analizados paso a paso

DIV DEVELOPER

Ensamblador, C, algoritmos... Todos los lenguajes de programación para que aprendas a desarrollar tus juegos

SOFTWARE ESPAÑOL

La más candente actualidad lúdica: Balance y Hammer, dos compañías volcadas en el videojuego español



CURSOS DE PROGRAMACIÓN

POR FIN LLEGA DIV 2

AHORA CON OPCIONES 3D

En Prensa Técnica hemos dado con la FORMULA

$\sum \frac{\text{inf. + act. + lin. + NT + imag. + lab. + hard. + soft. + Andros}}{264} = +PC!$

+PC



videojuegos



multimedia



hardware



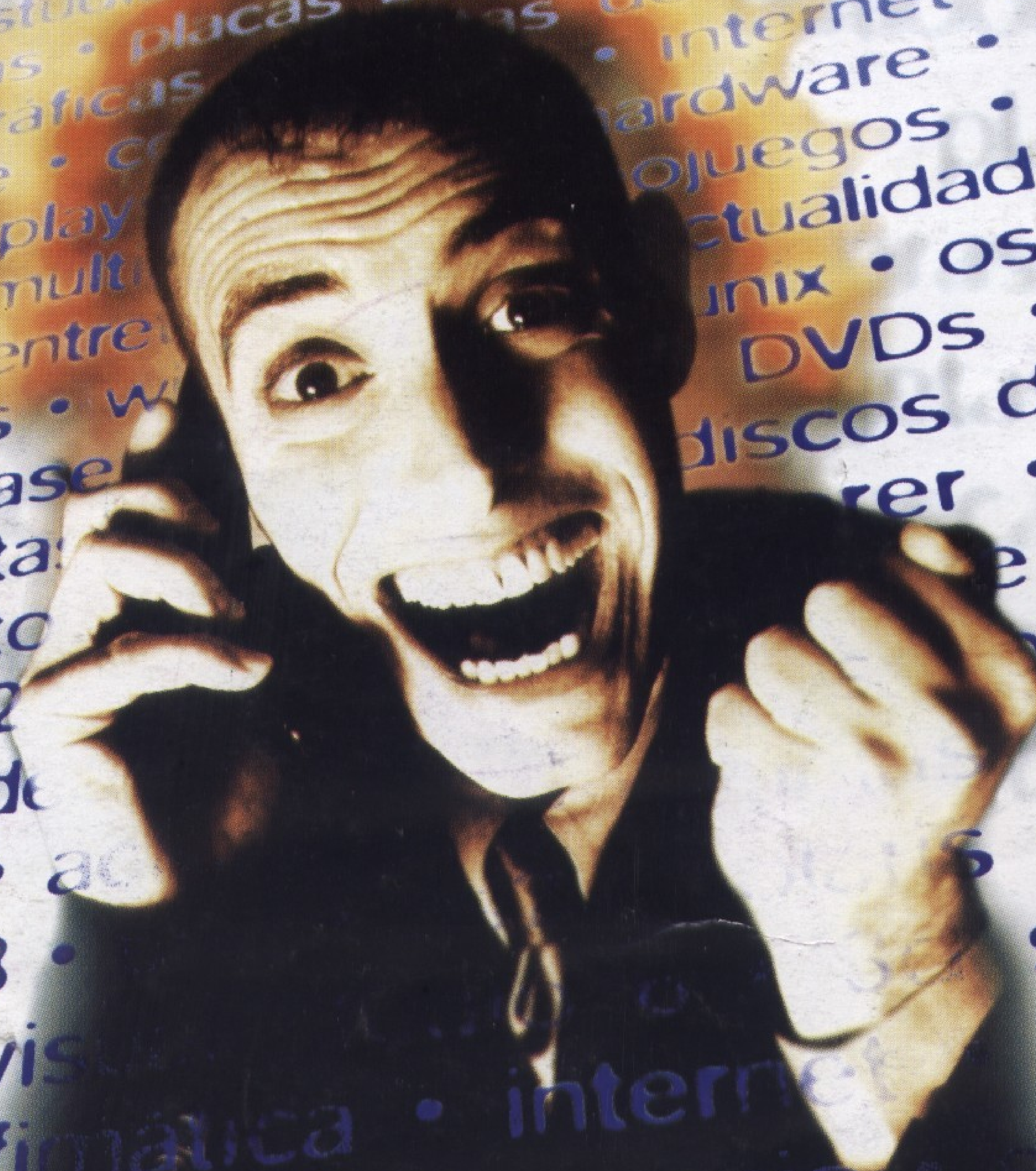
seguridad y virus



linux



ocio



* Más de 250.000 lectores nos avalan cada mes

¡Basta de llamadas! Con lo mejor(*) de nosotros hemos hecho la revista que lo reúne TODO para TODOS

+PC



+PC

(más)

La revista que te da MÁS

Anunciado en TV

Encontrarás todo lo que estabas buscando en más de 260 páginas de actualidad, reportajes, análisis, comparativas, opiniones, novedades del mercado y avances de todos los próximos lanzamientos.

MÁS CONTENIDO
Más información
Más formación
Más entretenimiento

DISEÑO
Photoshop Vs Painter
Creación de caricaturas
Ayer y hoy de las 3D

INTERNET
Iddeo
Comercio electrónico
Utilidades gratis para tu PC

DOBLE CD-ROM
- Curso de Office interactivo
- Demo de DIV GAMES STUDIO
- RED HAT LINUX

La polémica del estandar MP3. Música gratis en Internet. Lista de todos los WEBS prohibidos



La revista que te da más

www.prensatecnica.com

Año 1 • Nº 1 • 995 ptas.

2CD Rom

CD - ROM 1
- DIV Games Studio
- Curso interactivo de Word
- Demo jugable Tomb Raider III

CD - ROM 2
- Euriec, distribución Red Hat Linux en castellano
- Suite Star Office

260 páginas

MPMAN
El walkman para el sistema de audio MP3

COMERCIO ELECTRÓNICO
Nos anticipamos al Euro y al 2000

ADEMÁS

+ INTERNET
Iddeo, el servidor de Retevisión
BlueTooth, redes Lan sin cables

+ OCIO
Juegos en red, busca tu adversario
Historia del videojuego en España

+ DISEÑO
Desfiguramos la cara de Bill Gates
Photoshop 5 vs. Painter 5

+ SOFTWARE
Office 97 en fichas prácticas
Todos los lenguajes de Internet

+ LINUX
Comparativa de distribuciones
Eurielec, Linux en castellano

+ HARDWARE
Pantallas planas Lcd
Palm Tops

Los accesorios y el software para hacer de tu PC una supermáquina

MMX2
Diez razones por las que debes migrar


Los mejores DVDs del mercado


Micrófonos, Webcams y accesorios multimedia


Joysticks, volantes, rolsticks, pads, etc.


Periféricos USB, discos duros Microdrive

REPORTAJES
- Los procesadores que vienen
- DVD's, monitores, impresoras
- Nuevas tecnologías

SOFTWARE
- Sistemas operativos
- Suites de ofimática
- Herramientas de programación e infografía

DOSSIERS
- Nuevos procesadores
- Periféricos USB
- Telefonía móvil

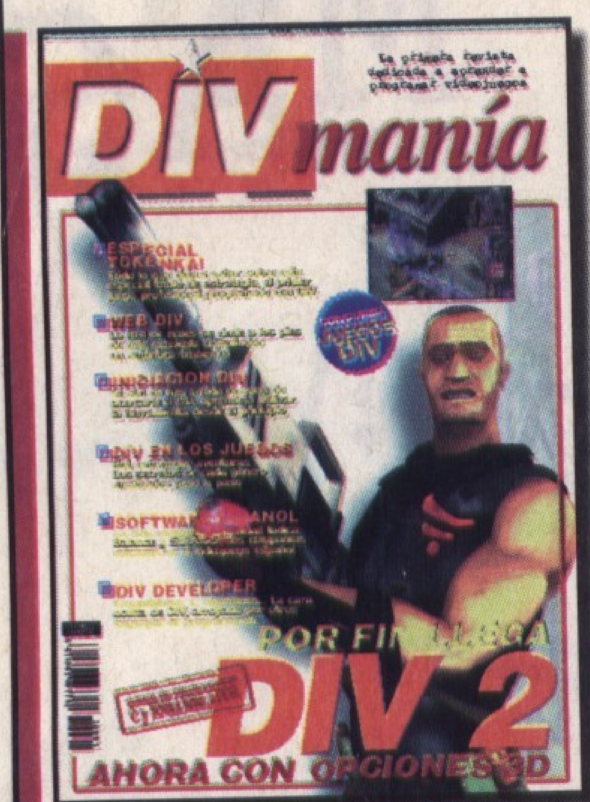
ACCESORIOS
- Joysticks, pads, volantes
- Webcams
- Micrófonos

Prens@ Técnic@
Edita PRENSA TÉCNICA
Alfonso Gómez, 42. Nave 1-1-2.
28037 Madrid
Tf: (91) 3.04.06.22
Fax: (91) 3.04.17.97

No te pierdas el número 1 de +PC

A la venta el **15 de Noviembre** sólo por 995 ptas.

DIVmanía

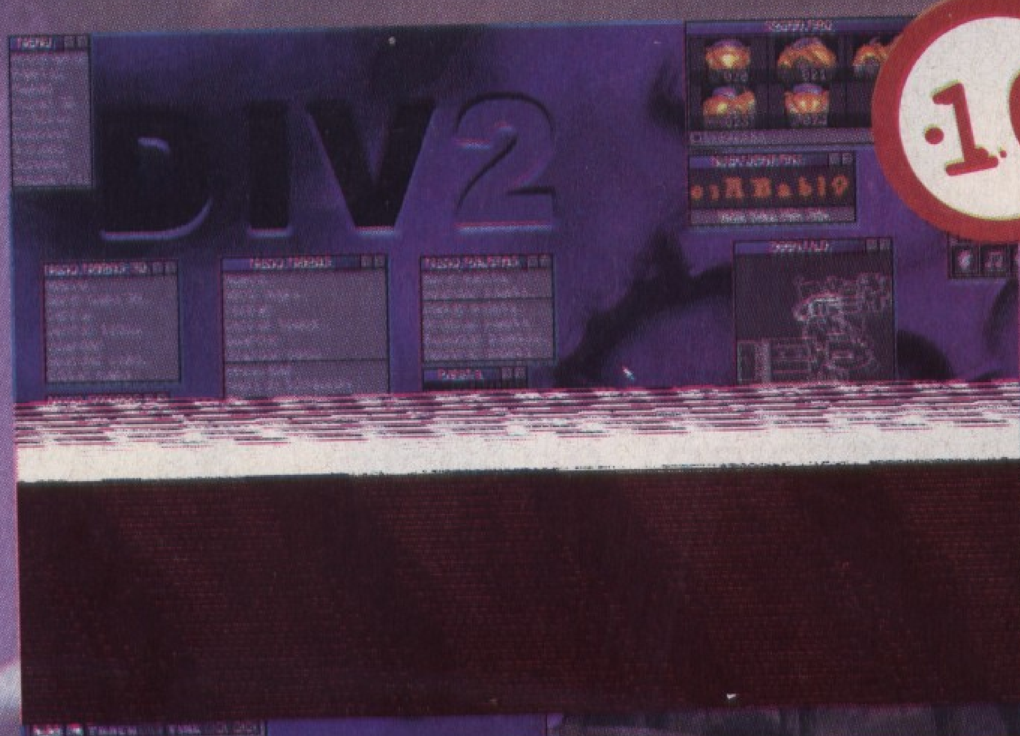


¿Amas la programación de juegos...? Léete estas líneas

Nunca imaginé que programar juegos llegara a ser tan fácil. Cuando me enganché a Atic Atac, me pregunté cómo se podría llegar a saber tanto de programación. Más tarde me decidí a comprar revistas llenas de código; lo tecleaba sin saber exactamente qué tenía entre manos y casi nunca funcionaba. Quince años después de mis primeros pinitos, la programación de videojuegos no tiene nada que ver con aquellos primeros tiempos, excepto que sigue siendo un apartado que casi está reservado sólo a privilegiados. El "casi" lo pone DIV Games Studio, ya que es una herramienta capaz de conseguir que casi cualquiera pueda desarrollar sus propios juegos.

Por este motivo empezamos por DIV para aprender a programar. Es, hasta el momento, la herramienta más sencilla. Pero DIVmanía está orientada a aquellos que quieran aprender a programar videojuegos y por ello abarca todos los aspectos del desarrollo. DIVmanía tendrá apartados para todos: noticias, comentarios de juegos de lectores, cursos de programación en DIV y otros lenguajes como C o ASM, grafismo, diseño y todo lo necesario para desarrollar juegos.

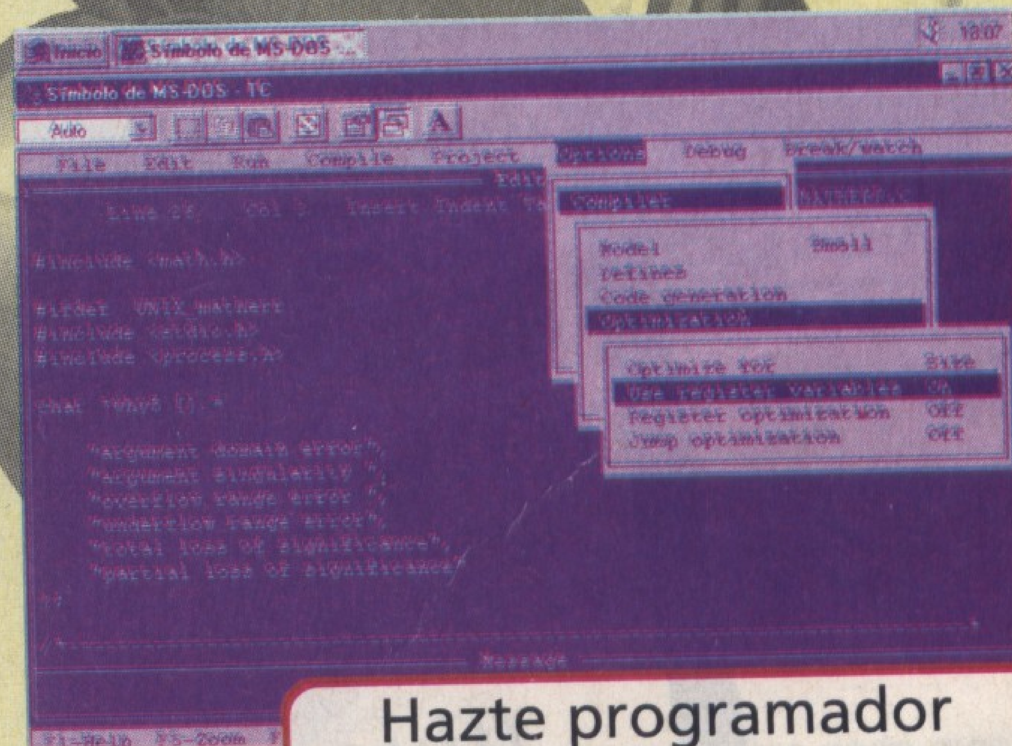
Os damos la bienvenida a todos. Os dedicamos un proyecto fruto de nuestra ilusión y la vuestra, que aparecerá cada dos meses en los kioscos. No quiero irme sin recomendaros que veáis la página de suscripciones, con innumerables ventajas. Estamos abiertos a todos vuestros comentarios y sugerencias. Podéis enviarlos a la siguiente dirección: Correo DIVmanía, C/ Alfonso Gómez 42, Nave 1-1-2, 28037, Madrid, o bien al E-mail de la revista: divmania@prensatecnica.com.


Interfaz de DIV 2



Estrategia isométrica

DIV Developer


Hazte programador

Director: Mario Luis
mluis@prensatecnica.com
Coordinador Técnico: Rafael María Claudín
divmania@prensatecnica.com
Colaboradores: Antonio Marchal, Pablo Trinidad, José M. Sevillano, Sergio Cánovas, Miguel Barroso, David Martínez, Emilio Llanos.
Edición: Julio Crespo, Daniel Izeddin, Ignacio Pulido, Eva María-Villanueva
Dirección de Arte: Francisco Calero

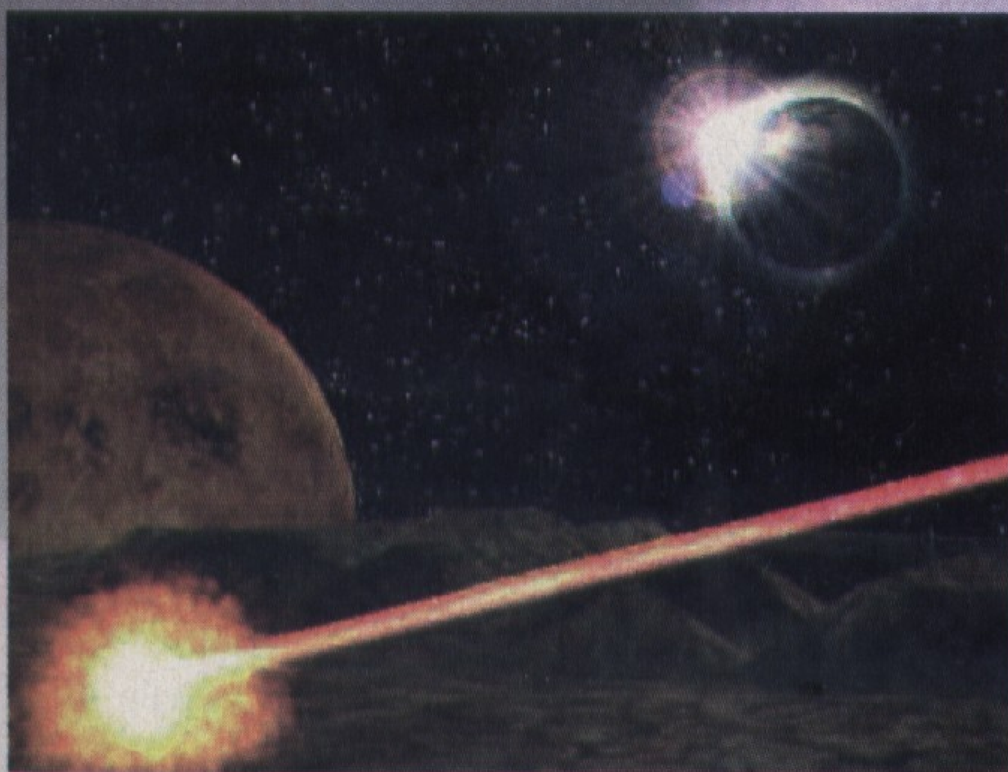
Jefa Dpto. Maquetación: Carmen Cañas
Maquetación: Marga Vaquero, Manuel J. Montes, Silvia M. Villanueva
Portada: Francisco Calero, Carlos Sánchez
Publicidad: Marisa Fernández, marisa@prensatecnica.com
Sonia Glez.-Villamil, sonia@prensatecnica.com
Susana Gómez, susana@prensatecnica.com

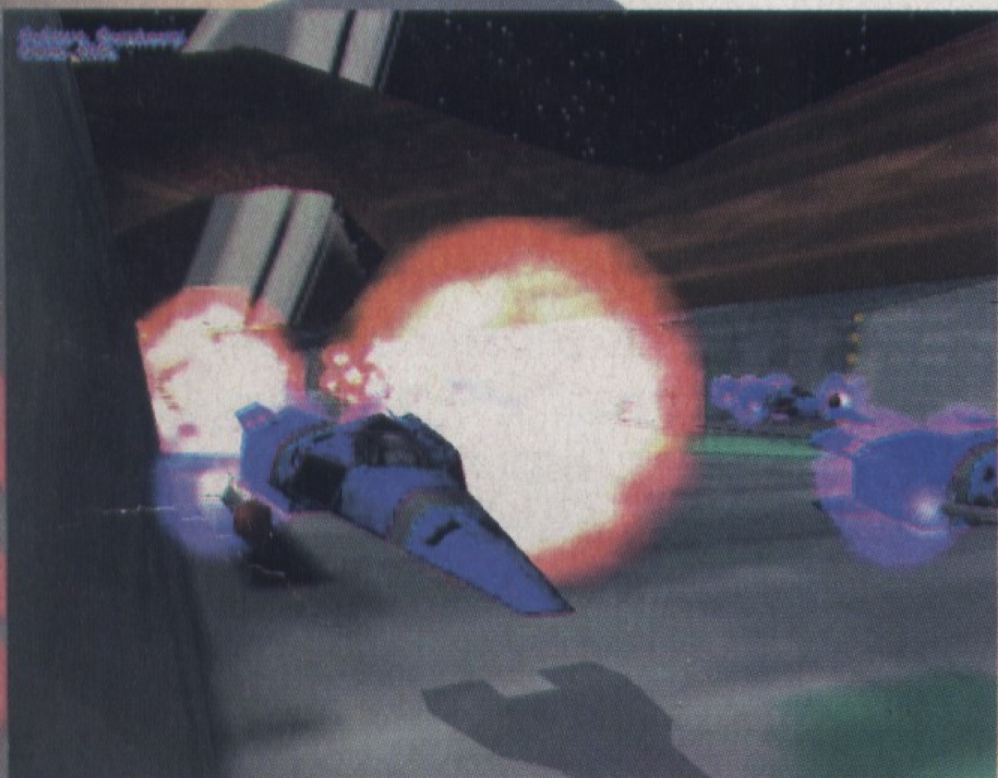
Supervisión CD-Rom: Jesús Fernández Torres
Servicio Técnico CD-Rom: David Amaro
Horario de atención: tardes 16:00 - 18:00 h
E-mail: stecnico@prensatecnica.com
Secretaria de Redacción: Eva Cascante
Departamento de Suscripciones: Sandra Fernández
suscripciones@prensatecnica.com

Departamento de Administración: José Antonio Rivas, Mario Salinas
Departamento Comercial: Ana Guillemat

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
c/ Alfonso Gómez 42, Nave 1.1.2
Madrid 28037. España
Tfno: (91) 304. 06. 22
Fax: (91) 304. 17. 97

Sumario

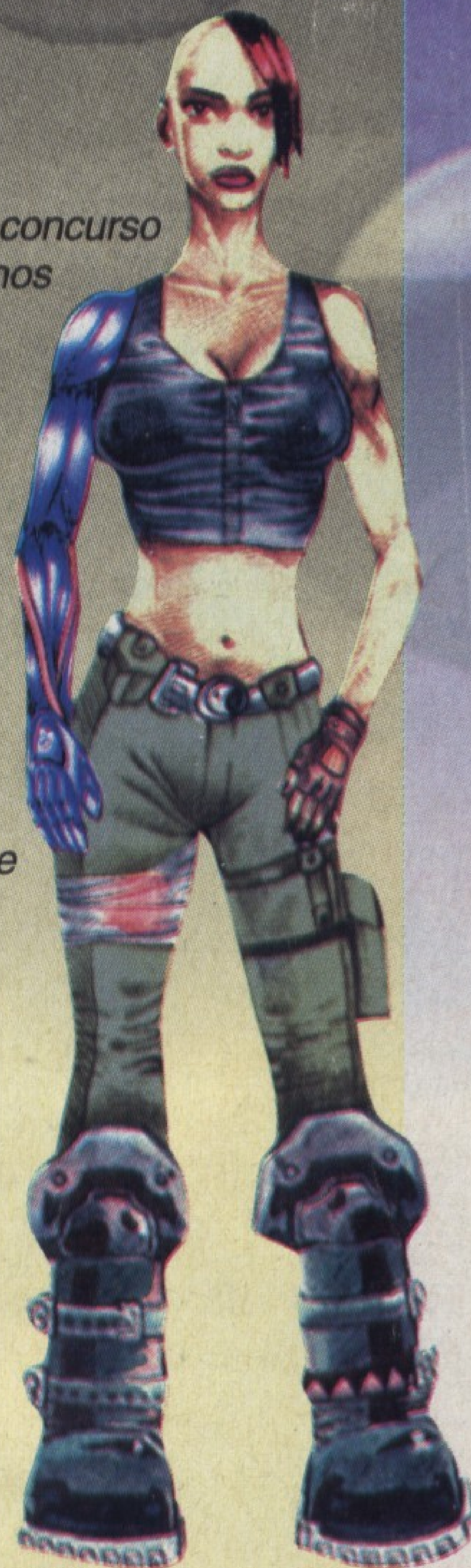








Programas del lector

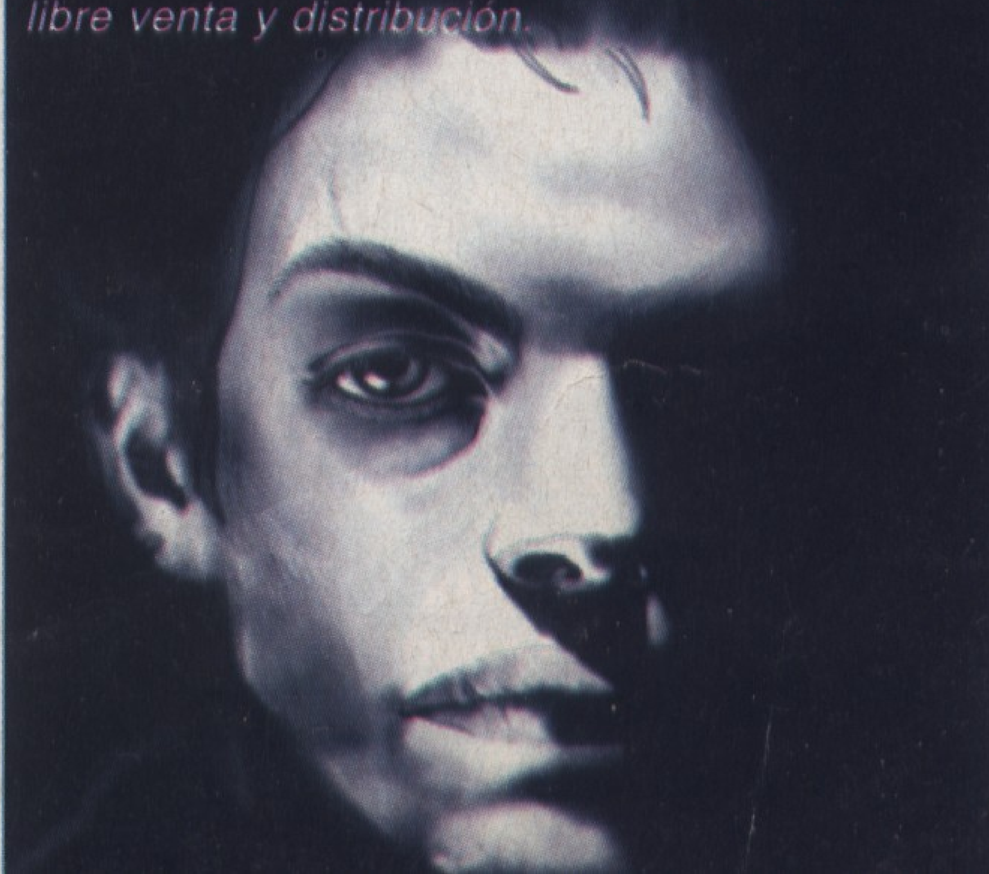


PREMIAMOS LOS MEJORES JUEGOS DE LOS LECTORES

DIV ha creado toda una escuela dentro del mundo de la programación. Todos los que se han acercado a la herramienta han sido capaces, de hecho, de programar. Por ello, realizamos un concurso entre los lectores que hayan realizado juegos con DIV. Os ofrecemos un primer premio de 25.000 pesetas y dos accésit de 20.000 pesetas.

¿QUÉ ES DIV GAMES STUDIO?

DIV Games Studio es una herramienta de programación que facilita en gran manera nuestra inmersión en el software de entretenimiento. Es el primer entorno profesional que permite realizar videojuegos con fines comerciales sin necesidad de un pago adicional. Con el carné de desarrollador incluido se permite el desarrollo de cualquier número de videojuegos y su libre venta y distribución.



Si llama desde fuera de España, marcar (+34)
E-mail: ntactual@prensatecnica.com
http://www.prensatecnica.com
Horario de atención al público: de 09:00 a 19:00 h ininterrumpidamente

EDITA: PRENSA TÉCNICA

Director General: Mario Luis

Director Editorial: Eduardo Toribio

Director de Producción: Jorge Rodríguez

Directora Financiera: Felipe Hernandez

Directora Publicidad: Marisa Fernández

Director Comercial: Esteban Martínez

Fotomecánica: M y F

Impresión: Printerman

Duplicación del CD-Rom: M.P. O., Servicios Ibéricos, Grupo Cóndor

Distribución: SGEL. Avda Valdelaparra, 29 Alcobendas. Madrid

Depósito legal: M-42077-1998
AÑO 1 • NÚMERO 1



El lector lúdico

Sólo para aficionados a los juegos

Éste es un espacio de opinión para usuarios de DIV y lectores de esta revista. Necesitamos que todos participéis con ideas y comentarios.

Canal y juegos

Estimados colegas, DIVadictos(*) y DIVas(*), para todos los que estéis interesados en desarrollar el juego compartido de la ciudad ideal que propuse os comento lo siguiente:

Estoy desarrollando una página web con el contenido de la propuesta más detallado, con vuestras ideas para ayudar, planteamientos, código, pero he de decir que todavía tardará unos días en aparecer. Estoy esperando las vacaciones ya que el trabajo y otros proyectos en DIV no me dejan más tiempo... De momento, pero tranquilos, todo llegará. Aún necesito más apoyo de vuestra parte... o ¿no os gustaría cambiar algo del lugar donde vivís? Los interesados deberíais dejaros ver más. Organizaré las propuestas y serán mostradas en la web. Debéis tener en cuenta que la ciudad se ha de crear sin impeDIVmentos(*), ¿ok?

Para mis colegas chateros: algunas semanas no podré dejarme ver por el chat. Utilizad el e-milio que es casi gratis por cortesía de Telefónica.

!Deemo

Esta carta ha sido elegida para poner al día a los seguidores de DIV que no tienen conexión a Internet y no están informados de lo que se ha cocido sobre DIV en la Red de redes. Hagamos un viaje en el tiempo a la fecha de salida de DIV, hace ahora más o menos un año. Nada más salir el producto y crear la página oficial, llegaron cientos de e-mails. Se creó una lista de correo, en la cual alguien propuso crear un canal de IRC. Una vez con el canal funcionando, otra persona volvió a proponer una especie del canal. En esta página se está creando entre todos los usuarios del canal un juego compartido. Hay otras secciones, como lista de IRCeros, sección de tertulias, tablón de anuncios, etc. A partir de ahora, gracias a la revista, esperemos que tanto las distintas páginas como el canal de IRC se revitalice, dando un nuevo impulso a todo lo pendiente. Las direcciones son:

http://www.divgames.com
http://www.redbcn.com/canaldiv

Gracias a vosotros

Atrás quedan los días en que leía un anuncio donde se buscaban programadores para realizar el proyecto más ambicioso del software español. Largos se me pusieron los dientes de envidia de ver el resultado. Un producto excelentemente acabado. La presentación es buena y trasparente a la hora de intentar descubrir nuevas aplicaciones.

Estoy en constante contacto con el producto desde hace tres meses, así como con páginas web sobre el tema, las cuales proliferan cual setas en el campo. También estoy suscrito a la lista de correo. Lo cierto es que deseo información sobre la versión profesional de la que tanto se dice que saldrá en breve, y espero que todos esos cambios que prometen sean un elevado conjunto de aplicaciones gracias a vuestro sudor.

Ya hago mis cosillas en DIV, y ahora estoy metido en un proyecto de aventura gráfica de armas tomar (¿alguien conoce a un buen grafista que no esté a más de 200 metros de mi casa?).También me metí con la técnica *FILMATIÓN* y la exprimí en unas ... dos horas. Lo que pasa es que ahora me está exprimiendo ella a mí cuando me he metido a programarla con movimiento a través de ratón. Con teclas no me parecía muy decorosa, incluso me da vergüenza saber que aquellos juegos en dos colores que tanto gustaban, incluso a mí, se hacían de esa manera, ahora tan «alcanzable» por el gran público con conocimientos medios de programación (gracias de nuevo por esa posibilidad).

Leo la mayor parte de las publicaciones de Prensa Técnica y no me pierdo ninguna de las apariciones de software. Precisamente por eso, puedo ser algo crítico con vosotros también. Me gustan bastante y no tengo demasiadas quejas, excepto algo de Game Over. Doy mi voto a que en breve tiempo se realice una revista con el nombre «DIV actual» o algo así. No hace falta que me extienda en cuál debería ser su contenido, queda claro.

Sin más, espero que mis opiniones no vayan a parar a saco roto y que próximamente reciba noticias vuestras, ya sea en los quioscos o en mi casa (por «correo tortuga» o por e-mail).

Alejandro Dosaula
Barcelona

Lo primero pedirte disculpas personales por bombardearte el e-mail. Pero es que yo, como dice la canción, soy DIVero porque el mundo me hizo así... Ya en serio, gracias por tu fidelidad a nuestros productos. Como podrás comprobar cuando leas esta revista, verás que tus ideas y sugerencias no caen en saco roto. Las críticas las hemos hecho llegar a las personas adecuadas (están pagando por ello).

(*) DIVadicto, DIVo, DIVa: Dícese de aquel aficionado a la programación de videojuegos que se encuentra fascinado por DIV.
ImpeDIVmentos: En lenguaje de DIVos, se trata de los problemas a los que se enfrenta un desarrollador que use DIV.

Programación avanzada en DIV

DIV GAMES STUDIO



Internet OffLine
www.divgames.com

UN LIBRO QUE PROFUNDIZA EN LA HERRAMIENTA

Ante la inminente aparición de DIV 2, la programación por medio de esta ya imprescindible herramienta de trabajo se dirige hacia muy altas miras. Es por ello que aparece este libro, que pretende servir de ayuda a quienes tengan intención de realizar un trabajo serio con ella.

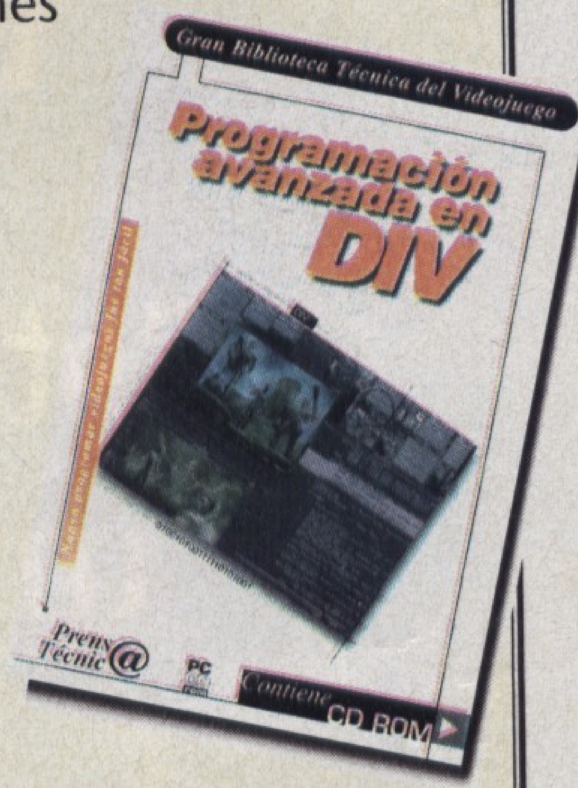
Para muchos programadores, grafistas y gente relacionada más o menos con la realización de videojuegos este compilador supuso un antes y un después. En muy poco tiempo hizo falta una ampliación de la información contenida en el manual original, debido principalmente al fenomenal éxito que tuvo el programa. Así nos llega ahora esta incursión en el DIV profundo.

No hay ninguna duda de que cualquiera que se acerque a DIV será capaz de programar, sea cual sea su nivel previo. Éste es uno de sus grandes atractivos. Se trata de un lenguaje extremadamente sencillo que permite obtener unos resultados increíbles sin demasiados quebraderos de cabeza.

El libro abarca todos los aspectos de DIV. En la primera parte podrás hacerte con la información teórica necesaria para enfrentarte al producto. Dentro de ella te ofrecen los primeros ejercicios prácticos con los que podrás empezar a soltarte con DIV Games Studio.

En el segundo apartado se entra de lleno en materia analizando las posibilidades de DIV con ejemplos concretos de diversos tipos de juegos, desde "matamarcianos" hasta programas de estrategia, pasando por plataformas o videoaventuras. Se estudian las necesidades de cada una de estas modalidades de juego y las enormes posibilidades que ofrece DIV para llevarlas a cabo.

En un tercer bloque del libro se presta atención al fenómeno DIV, con especial cuidado de la información que es posible encontrar en Internet al respecto. En efecto, a estas alturas hay ya un buen número de páginas web más o menos dedicadas a DIV Games Studio, así como canales de IRC, etc.



EL CD-Rom que acompaña al libro también es muy completo. Para empezar, te ofrece ejemplos de cuatro tipos de videojuegos: plataformas, "matamarcianos", aventura y estrategia. También te permite navegar off-line por algunas de las páginas dedicadas a DIV. Por último, han incluido utilidades shareware obtenidas en la Red de redes que pueden servir de gran ayuda a todos los usuarios.

Sir-Tech se viene abajo

EL FIN DE UNA VETERANA COMPAÑÍA

Una casa con una larga tradición en el mundo del software lúdico va a cerrar. Una mala noticia para quienes nos encontramos dentro de este mundillo. Los grandes de la industria se están empezando a comer a las pequeñas casas programadoras, que no pueden competir ni contra el fuerte volumen de trabajo ni contra las superproducciones lúdicas. Todos los que conocíamos a esta compañía desde hace unos cuantos años acogemos la noticia con tristeza. Sin embargo, tenemos que señalar que ya se ha anunciado que van a finalizar alguno de los trabajos pendientes de la compañía, entre ellos los esperados *Jagged Alliance 2* y *Wizardry 8*.

Hammer Technologies amplía sus miras



Hammer, una de las más importantes casas programadoras españolas, está creciendo a marchas forzadas con el fin de continuar con su racha de éxitos. No hay que olvidar que esta sólida compañía, responsable de la herramienta de programación DIV que ha dado origen a esta revista, tiene en preparación, aparte de DIV 2, productos como *Tokenkai*, un programa de estrategia realizado con DIV, *Jagdverband 44*, *Screaming Eagles*, un fenomenal simulador de vuelo de época, y *Neon Angel*, un plataformas 3D que hará frente a los primeros del género.

Cyberrumores

LA WEB NOS CUENTA SUS SECRETOS

Hablemos en primer lugar de *Simon The Sorcerer* y una problemática tercera entrega. La compañía responsable del producto, Adventure Soft, ha vivido en los últimos tiempos malos momentos que han llevado a una reestructuración de su plantilla. Sin embargo, el grupo de programación que tenía bajo su cetro esta saga, Headfirst Productions, tiene intención de continuar con ella. Lo que no está claro todavía es cuál va a ser la orientación del programa. Podría convertirse, por una parte, en una típica videoaventura al estilo de *Monkey Island*, como ya sucediera con anteriores versiones de *Simon*. Por otro lado, quizá acabe como un nuevo *Tomb Raider*. Quién sabe. Lo que sí es seguro es que uno de los responsables del producto ha pedido la opinión de los usuarios al respecto...

Por otra parte, se dice que el prestigioso Steven Spielberg quedó impresionado por el apartado sonoro del programa *Trespasser*, que se está desarrollando en su compañía DreamWorks. Al parecer, pasó por los estudios donde se está realizando el proyecto y pronunció palabras elogiosas para el compositor de la banda sonora del juego. Y es que los resposables del apartado sonoro del juego son los estudios SounDelux, que se han encargado de películas como *Mentiras arriesgadas*.

Los usuarios hacen el nuevo Quake

ID COMERCIALIZARÁ UNA ACTUALIZACIÓN CON NIVELES DE USUARIO



Una buena noticia para los fanáticos de *Quake II* que han puesto su granito de arena realizando niveles para el programa. Todavía tenemos fresco el caso de 3D Realms y *Duke Nukem 3D*. Por si alguien no se enteró, 3D Realms ganó una demanda a MicroStar por vender mapas para el juego realizados por los usuarios sin el consentimiento de la compañía. Pues bien, id Software ha decidido seguir otra política anunciando el lanzamiento de una actualización para *Quake II* realizada con los mejores mapas que hayan creado los usuarios y MODs que circulan en la actualidad por la Red de redes. La única pega que le pondríamos al asunto es que no pagaran a los autores de los niveles... Aunque no sabemos cómo lo harán.





Pronto podrás sentir el terror

BLACKSTONE, PRÓXIMA VIDEOAVENTURA DE JOHN SAUL



Excelentes programas se nos están acercando en estas fechas. Comentaremos aquí uno de ellos, la excelente videoaventura de terror *John Saul's Blackstone Chronicles*, que muy pronto presentará entre nosotros Proein, S.A. Se trata de un programa basado en una serie de relatos del conocido autor John Saul, que ha preparado especialmente para acabar como videojuego. Sin embargo, las historias han sido publicadas en su país con considerable éxito. El programa se ambienta en un pueblo misterioso en el que hay un viejo asilo del que parten las más extrañas historias. La calidad gráfica que nos ofrece el juego es muy elevada en todo momento, y esperamos que aparezca con las voces y los textos traducidos a nuestro idioma.





Ubi Studios programará en España

UN NUEVO GRUPO DE DESARROLLO SE SUMA AL RESURGIR DE NUESTRO PAÍS



Un nuevo grupo de programación ha aparecido en fechas recientes dentro de nuestras fronteras. Apadrinados por la compañía Ubi Soft, una importante distribuidora que nos ha llegado del país de los galos, los chicos de Ubi Studios van a intentar ponerse al nivel de un mercado que en estos momentos se encuentra en plena forma, como demuestran los últimos lanzamientos patrios. En la fotografía que acompaña esta reseña se puede ver a algunos de los componentes del grupo que muy pronto pondrá productos en circulación en el mercado.

Crta. de Rubí 72-74, 3ª plta.
Edificio Horizon
08190 Sant Cugat del Vallés
(Barcelona)
E-Mail: mantras@ubisoft.es

EXCLUSIVA

DIV 2, la nueva revolución

El compilador incorpora nuevas opciones 3D

Cuando salió al mercado DIV, fue en cierto modo una revolución. Ahora va a salir a la venta DIV 2. Se han arreglado fallos y *bugs*, se han mejorado algunos aspectos del DIV anterior, y además se han añadido unas nuevas posibilidades, que abren todo un abanico de creación y diseño.



Un nuevo y cuidado *look* para la segunda edición de la herramienta.

Antes de empezar, hay que reseñar que en este nuevo DIV 2, se han solucionado gran cantidad de errores, o como se conocen comúnmente *bugs*. Esto en parte ha sido posible gracias a la ayuda de todos vosotros. También nos han llegado gran cantidad de sugerencias para el nuevo DIV. Dentro de lo posible, hemos intentado tener en cuenta todas vuestras sugerencias, considerándolas siempre en función de su utilidad.

La magnífica respuesta de los usuarios ha motivado la aparición de una seguida edición mejorada de la herramienta.

Dentro de los pequeños errores más conflictivos, tenemos que referirnos a todo el tema del tratamiento de ficheros FPG. Antes no era muy fiable, los fallos que se producían nos llevaban a situaciones difíciles. Todo este tema está solucionado, es más, se han implementado mejoras que tienen que ver con dichos ficheros y que iremos viendo detenidamente. En cuanto a otros errores apreciables o que se puedan reseñar, está solucionado el tema VESA, dando una total compatibilidad, incluso en resoluciones bajas. A aquellos a quienes haya ocurrido, sabrán de qué estamos hablando. Por otro lado, en cuanto a resolución de *bugs*, se han solucionado algunos más que no son reseñables, siendo estos dos temas los más importantes.

Pero vamos a dejar de hablar del DIV antiguo, como pronto se le conocerá. Comencemos a ver las nuevas prestaciones de DIV 2. El trabajo ha sido arduo y lo continúa siendo. Existirán aquellos que se hayan encariñado con el anterior DIV, pero como dice un dicho popular, "A Rey muerto, Rey puesto". Así que sin mas dilación, empecemos a hablar de la nueva *criatura*.



La librería de gráficos sigue siendo enorme.

Arrancando

Ha habido muchas preguntas sobre la salida de DIV 2. La fecha confirmada ronda el mes de noviembre, ya que seguimos trabajando con tesón. De todas formas os informarán con seguridad de su fecha exacta de salida mediante anuncios publicados en diferentes revistas, también encontraréis entonces su precio final de venta al público.

Las mejoras ya incluidas en DIV 2 no son pocas, aunque no nos cansamos de recalcar que seguimos trabajando. Existen ciertos temas que, aunque están en preparación, son seguros. Aun así, debido al gran numero de sugerencias, no nos ha sido posible incluirlas todas, pero seguiremos en la labor. Algo que se deja notar a primera vista es la optimización del código de DIV y el intérprete para funcionar sólo con Pentium. En parte se ha tenido que tomar la decisión de si seguir dando apoyo a ordenadores como el 486, 386 e inferiores. Se ha decidido que no, ya que además de existir pocos ordenadores con estas características, el producto debe avanzar y adaptarse a los nuevos mercados.

Pero ahora vamos a comenzar con las nuevas mejoras del entorno DIV. Existen ciertos cambios dentro del mismo que saltan a la vista, nada más cargar el programa. El primero de ellos es el diálogo inicial. Éste ha cambiado, ahora se podrá registrar la copia del programa. Además se podrá cargar el entorno de dos maneras distintas en memoria. Una guardando el

trabajo anterior y continuando con el escritorio tal cual lo teníamos. Y otra que nos permite reiniciar el entorno, borrando todo el escritorio de una pasada. Es útil en algunos casos, donde el caos del escritorio sea tal que la mejor decisión sea limpiarlo y empezar de nuevo.

Otro detalle importante es el de los listados coloreados. Cada función, variable, número o signo aparecerá de un color distinto. Esto mejora la comprensión de los listados y facilita su lectura. Además, podremos cambiar los colores que anteriormente eran configurables, como eran los del fondo de los listados. Estas opciones son configurables desde el menú *Sistema*, como la vez anterior. También han incorporado cosas como la selección del tamaño del tabulado, uniendo todo ello hacen del editor una herramienta más configurable.

Y hablando de opciones elegibles por el usuario y que tienen que ver también con el entorno, algo que también ha cambiado, en cuanto a forma de elegirse, es el tapiz de fondo. Se hace uso de una gama de colores, para así poder convertir y visualizar el fondo de pantalla de multitud de formas. Este nuevo uso de gama de colores, también se podrá ver en acción en otras partes del programa. Ahora, con la nueva forma de uso, se puede crear una gama personalizada, hasta tal punto, que podemos elegir todos y cada uno de los colores que la componen.



Es más sencillo controlar los mapeados.

Otro aspecto que ha recibido grandes cambios es el de la ayuda. Se han incorporado botones, para facilitar la navegación por las páginas. Ahora tiene cierto parecido

Entrevista con Daniel Navarro, el padre de la *criatura*

La idea de DIV surgió de una única persona, su nombre, como muchos sabréis es Daniel Navarro. Nos ha prestado algo de su tiempo, lo que sin duda agradecemos, ya que no dispone de mucho, pensad que está trabajando a toda maquina en DIV 2. Así que aquí está la tan prometida entrevista:

DIV MANIA: ¿Cuál ha sido tu objetivo al diseñar esta segunda versión de DIV?

DANIEL NAVARRO: *Pues intentar completar DIV, en general, como un entorno de desarrollo de juegos. Hace un año, cuando estábamos acabando la primera versión, nos faltó, sin ninguna duda, tiempo para poder incluir un montón de ideas. Además hemos recibido constantemente desde entonces cartas y E-mails con problemas que se han encontrado los usuarios y con sugerencias. Todos los problemas se han solucionado, pero lo realmente difícil ha sido seleccionar cuáles de las sugerencias debíamos incluir primero. No todo el mundo estará de acuerdo al 100% con la selección de mejoras incluidas, pero deben entender que han existido muchos factores determinantes en esas decisiones, como por ejemplo el tiempo que requería por nuestra parte, o lo bien que esa idea podría integrarse con el resto de componentes en DIV.*

DIV: ¿Crees que se han añadido suficientes novedades como para justificar esta nueva versión del programa?

DN: *Si, claro, por supuesto, sin ninguna duda, ¿te parecen pocas? En un principio se planeó sacar la segunda versión como DIV Profesional en julio de este año, después septiembre y finalmente saldrá para Navidades. El motivo es que no hemos querido sacar una nueva versión sólo con alguna cosa más, sino que se ha pretendido dar un repaso completo para mejorar DIV, aunque sólo sea un poco, pero en todos sus aspectos. Se ha tratado de avanzar en dos aspectos, hacer DIV más fácil y cómodo de utilizar y, por otro lado, hacerlo más potente.*

DIV: ¿Cuáles crees que son las novedades más importantes?

DN: *Hay varios aspectos importantes, yo distinguiría entre las novedades del propio entorno y las del lenguaje de programación, ambas se han trabajado por igual. Es difícil decidir cuáles son las novedades «estrella», no lo sé, quizá todo el aspecto más visual de DIV, quizá el modo 8 o la posibilidad de hacer juegos por red, los módulos de sonido, las funciones de búsqueda de caminos, las nuevas herramientas (como el editor de sonidos, calculadora, el generador gráfico), las mejoras en el lenguaje como tablas y estructuras multidimensionales, etc, ciertamente no lo sé.*

DIV: ¿Y cuál ha sido el aspecto que más esfuerzo o dolores de cabeza, os ha traído?

DN: *Coordinar los esfuerzos de toda la gente que ha intervenido, cuidando que nadie saliera malherido en su salud mental e intentando mantener la moral bien alta. El maldito modo 8, con su mapeador integrado nos sigue, a día de hoy, volviendo locos con sus problemas. No ha sido fácil casi nada en este desarrollo, la verdad es que hemos tenido que trabajar duro para ganarnos el pan.*



Daniel Navarro

DIV: ¿Cuál ha sido exactamente tu función y cuánta gente ha intervenido en este trabajo?

DN: *Yo llegaba por la mañana, cogía mi látigo castigador y ¡ale!, a currar todo el mundo. He sido el que decidía sí o no en prácticamente todos los aspectos del desarrollo y, en fín, que no sé cómo todavía me quedan amigos, quizá finjan porque me temen. Dejando las bromas a un lado (sí, sí, bromas) la verdad es que hemos sido tres programadores principales (Juanjo Garrido, Luis Fernando Fernández y un no muy humilde servidor) otros tantos grafistas (Carlos Cabañas, Rafael Barraso y Miguel Ángel Carrillo) y, como ya viene siendo habitual, un montón de colaboradores.*

DIV: Y ya por último, ¿hay algo que te gustaría comentar o explicar a todos los lectores?

DN: *Sí, que en el cierre de esta segunda versión nos ha seguido faltando tiempo. Parece que no importa lo que hagamos, nunca nos da tiempo a poner todo lo que queremos. DIV 2 es mejor y tiene más cosas que DIV, pero sigue sin tenerlo todo, estoy comenzando a sospechar que es imposible tenerlo todo.*

DIV: Gracias por tu atención y que sepas que estamos todos impacientes con la aparición de DIV 2. Ahora, te dejamos que sigas trabajando.

DN: *¿Que decías? ¡ah!, sí, gracias a vosotros.*



Nuevo editor de sonidos con la interfaz mejorada.



Pantalla de manejo de los gráficos.



Así se nos presenta el nuevo DIV.

con algunos navegadores de Internet que todos conocemos. Esto le da una accesibilidad muy alta. Además podemos imprimir cualquier página de la ayuda. También se han incluido gráficos en la ayuda, dando la posibilidad de que éstos sean a 256 colores. Otro tema que tiene que ver con la ayuda es que ahora cuando se accede a la lista de procesos, mediante la pulsacion de la tecla F5, se preselecciona el proceso que se encuentre bajo cursor en el programa. Pero sigamos hablando de las copias de papel y en lo que se refiere a imprimir, se han incluido opciones para poder ver los listados de forma impresa. Y no sólo eso, también podremos sacar una copia por impresora de los contenidos de los FPG.

La fecha confimada de lanzamiento se ha fijado en noviembre, ya que seguimos trabajando con tesón

Los ficheros FPG, también han cambiado drástricamente, por lo menos en el tema que tiene que ver con el entorno. No os preocupéis, los ficheros serán compatibles. El cambio reside en que ahora, a la hora de visualizarlos dentro del entorno, aparecerán más datos. A la hora de incluir un FPG, éste requerirá un nombre y una descripción, de la misma forma sucederá a la hora de manejarlos. También se van a incluir para exportar e importar mapas gráficos automáticamente en dichos ficheros FPG. Esto ahorra mucho trabajo, ya que a un golpe de ratón conseguiremos construir un fichero FPG a partir de un mapa gráfico y viceversa. Luego a la hora de manejarlos existirá una opción con la que podremos ver de forma gráfica, mediante representación en miniatura los gráficos contenidos dentro del FPG.

Este tipo de *browser*, que es como se denomina comúnmente a los visualizadores de gráficos, ha sido implementado en otras partes del entorno. Cuando carguemos un fichero con un mapa gráfico del tipo MAP, PCX, etc., también aparecerá una imagen representativa del gráfico antes de cargarlo. Así, de un solo vistazo, sabremos todas las imágenes que tenemos en el disco duro. No olvidemos que ya desde dentro del propio DIV se pueden leer nuevos formatos de imagen. Se están intentando incorporar el máximo de estos formatos. Pero sin irnos del tema, no podemos olvidar decir que los *browser* de los que hablábamos se han incorporado también a la carga de ficheros de sonidos, siendo posible escucharlos antes de cargarlos en memoria, además de representarse gráficamente. Aparte, existe la posibilidad de cargar en memoria varios ficheros a la vez. Esto simplifica mucho el trabajo, ya que cuando se tienen muchos ficheros, por ejemplo 100, es una ardua tarea cargarlos uno a uno en memoria.

Y hablando de sonidos, dentro del entorno se va a incluir también un editor de sonidos. Con él podremos hacer algunos arreglos a los sonidos que grabemos o cojamos de alguna librería. Se podrán hacer fades de distintos tipo, cambiar volúmenes, reducir ruidos y en definitiva, todos esos efectos de los que disponen los editores de sonido. También dentro del aspecto sonido, ahora se dispone de una nueva ventana que controla los distintos volúmenes de la música, el canal de sonido y el máster. Con estos cambios, queda bastante tratado el tema del sonido. Aunque existen también mejoras dentro de este tema, pero ya con más vistas al lenguaje de las que hablaremos más adelante.

Ahora pasaremos a hablar de una herramienta muy versátil y que además es una novedad. Se trata del generador de sprites y lo que exactamente hace es que nos permite generar animaciones de muñecos en unos breves segundos. Tiene varias opciones configurables, como la animación a realizar, como puede ser andando, corriendo, etc. Además, debemos elegir el modelo de muñeco, entre hombre, mujer o niño. Por último, únicamente aplica una textura que es un dibujo de la piel, el vestido y demás. Y el DIV se encargará de juntar todo y sacar una animación con el número de imágenes que queramos. También se puede seleccionar el ángulo de visión. Esta herramienta será muy útil, tanto en juegos 3D, como 2D, ya que cambiando algunos de los parámetros a seleccionar conseguiremos ahorrarnos mucho trabajo gráfico.

Existen algunos otros temas dentro del entorno que si no se están haciendo en este momento, se harán en breve y por lo tanto se pueden



La nueva interfaz de mapas es más intuitiva.

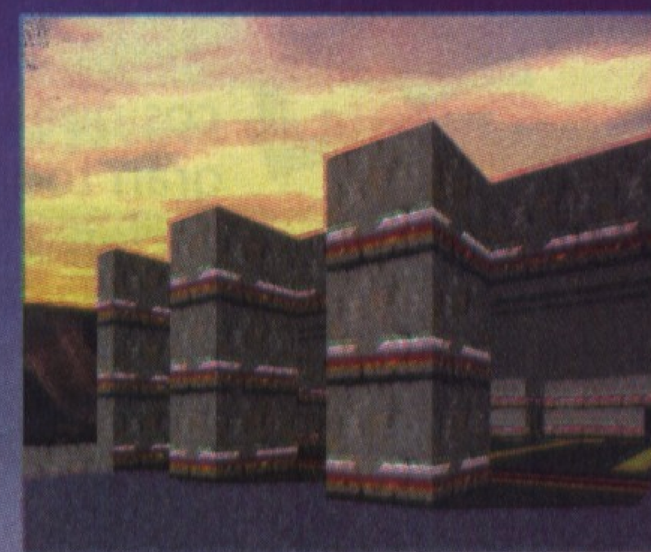
Hazte Programador

DIV









Queridos lectores y aficionados a la programación de videojuegos: desde este momento, y de forma totalmente gratuita, podéis entrar a formar parte de un nuevo y especial Club DIV, realizado por esta revista que pretende convertirse en la niña de los ojos de todos vosotros. En esta misma página encontraréis un carné de Programador de DIV. No queremos decir que con él vayáis a entrar en una logia programadora con la que se os abrirá la mente y de golpe seáis capaces de programar programas como TokenKai. Sencillamente, pasaréis a disfrutar de alguna de las ventajas que os ofrecemos, y de forma totalmente gratuita.

Para tener este carné de Programador de DIV no tenéis más que rellenar el que viene algo más abajo y enviárnoslo a nuestra dirección habitual:
C/ Alfonso Gómez, 42, Nave 1-1-2,
28037 Madrid, España
señalando Club DIV en el sobre. Entre las ventajas que podrás disfrutar, por el momento te ofrecemos información actualizada de los programas de la casa y, lo que es más importante, un 10% de descuento en las siguientes actualizaciones de DIV. Además, sorteamos 10 suscripciones a DIV Manía y 10 videojuegos Snow Wave Avalanche. Los resultados de este sorteo aparecerán en el número 2 de la revista.

Carné de Programador



Nombre: ....

Dirección: .... **C.P.:** ....

Ciudad: .... **Edad:** .... **Teléfono:** ....

PERFIL DE TU EQUIPO: **Procesador:** ❑ 486 ❑ Pentium ❑ Pentium II ❑ Otro .... **Sistema Operativo:** ❑ MS Dos ❑ Windows 3.1 ❑ Windows 95 Windows NT ❑ Otro .... **Memoria RAM:** ❑ 8 ❑ 16 ❑ 32 ❑ Otro .... **Tarjeta Gráfica:** ❑ 1Mb ❑ 2 Mb ❑ 4 Mb ❑ Otra .... **Tarjeta Aceleradora:** ❑ 3Dfx ❑ Power VR ❑ Otra .... **Tarjeta de Sonido:** ❑ Sound Blaster compatible ❑ Gravis ❑ Otra .... **Lector de CDROM:** ❑ 2X ❑ 4X ❑ 8X ❑ Otro .... **Modem:** ❑ 14400 ❑ 33600 ❑ 55400 ❑ Otro ....

PERFIL DE USUARIO: **¿Para qué usas tu PC?** ❑ Juegos ❑ Programación ❑ Diseño ❑ Ofimática ❑ Otros

¿Dónde lo usas? ❑ Casa ❑Trabajo **¿Cuántas horas al día de media lo usas?** ❑ 2 ❑ 4 ❑ 6 ❑ Otros

¿Haces uso de Internet frecuentemente? ❑ Si ❑ No **¿Tienes acceso a una red local?** ❑ Si ❑ No

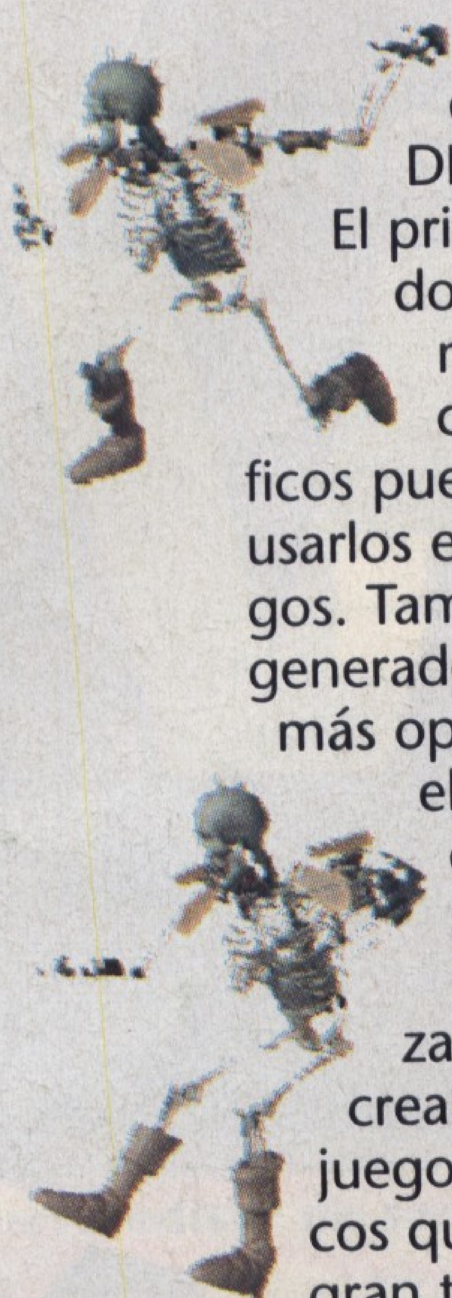
¿Posees una consola de videojuegos? ❑ Si ❑ No **¿Cual?**..

¿Donde has comprado este producto? ❑ Quiosco ❑ Tienda ❑ Centro Comercial ❑ Libreria ❑ Por Correo ❑ Otros

¿Por qué decidiste comprar el producto? ❑ Portada ❑ Precio ❑ Amigos ❑ Revistas ❑ Publicidad ❑ Tematica ❑ Otros

¿Te consideras buen...? ❑ Grafista 2D ❑ Grafista 3D ❑ Progr. Principal ❑ Progr. de Apoyo ❑ Músico ❑ Aprendiz

¿Qué revistas especializadas lees habitualmente?



dar por hechas dentro de DIV 2.

El primero de ellos es el generador de texturas, que genera mapas gráficos a partir de cálculos fractales. Estos gráficos pueden ser muy útiles para usarlos en paredes de nuestros juegos. También se quieren retocar el generador de explosiones, dándole más opciones. Por último y no por ello menos importante, se quiere conseguir un sistema para realizar instalaciones que puedan estar personalizadas. Es decir, a la hora de crear la instalación de nuestro juego, podremos elegir los gráficos que aparecerán, dándole un gran toque personal a nuestras creaciones. Además, las instalaciones ahora serán compactadas. Es decir, cuando deseemos hacer una instalación ejecutable de nuestro juego, se compactará todo en un único fichero. Este fichero ya no será accesible desde DIV, con esto se consigue que nadie retoque nuestros gráficos, a no ser que tenga los originales, claro.

Una vez ya dentro del editor gráfico, la mayor mejora es que ahora se pueden utilizar los mapas gráficos para más operaciones. Por ejemplo, podemos crear un mapa gráfico con una bola y utilizarlo para pintar con el pincel, tirar líneas, círculos, etc. Con esto se abre un abanico de posibilidades. También se le quieren incorporar posiblidades para facilitar la creación de mapas. Este tema estaría relacionado con la programación, ya que funcionaría simultáneamente. Se crearía el mapa y el DIV se encargaría de tratarlo internamente, *tileándolo*.

Y dentro del aspecto gráfico del entorno, ahora se ven mejor los títulos de las ventanas desactivadas (mapas, prgs, ficheros, etc.) además se le ha dado un aspecto degradado en la barra de título de las ventanas. Todo ello unido, hace que desde la primera ventana ya veamos los cambios de DIV 2.

Han incluido más cosas en el entorno. Disponemos de una calculadora en menú de sistema, con un evaluador de expresiones matemáticas completas. Es decir, se puede introducir paréntesis y crear operaciones de gran complejidad. Además, en todas las demás ventanas existe un nuevo control para mover, minimizar y cerrar ventanas, ahora se puede abortar. Y el movimiento por las barras de la ventana es más versátil y suave.

Los dos últimos temas tienen que ver con los gráficos. El primero de ellos es la importacion de ficheros con extensión JPG, BMP comprimidos y PCX y BMP de 16 y Truecolor. Ahora importaremos los mapas graficos desde el entorno y no tendremos que usar otros programas externos para la conversión de formatos y colores. El otro cambio está dentro del generador de *fonts*, ya que ahora hace antialias entre el *font* y su *outline*, creando unos aspectos para nuestros textos mucho más agradables a la vista.

Con esto quedaría hecho el boceto de las mejoras en cuanto al entorno. El siguiente tema merece mención aparte. Se trata de un nuevo modo de visualización dentro del entorno de programación. Ya tiene nombre, modos 8, para indicar el hecho de la ampliación drástica sobre los modos 7. Se trata de los mundos en 3D.

Mundos 3D

Todo el mundo ha jugado a juegos del tipo *Doom* y parecidos. En ellos nos movíamos por un castillo donde debíamos asesinar a gran multitud de enemigos. Los modos 8 toman este tipo de funcionamiento a la hora de visualizar mundos en 3D. Se dispone de un mapa de vista cenital en dos dimensiones. Gracias a dicho mapa se pueden pintar paredes, doseles o escalones dentro de nuestro mundo 3D. También se podrán pone sprites, como pasaba con los modos 7. Todo ello hace que en el nuevo DIV 2, el límite de los mundos virtuales esté dentro de nuestra imaginación.

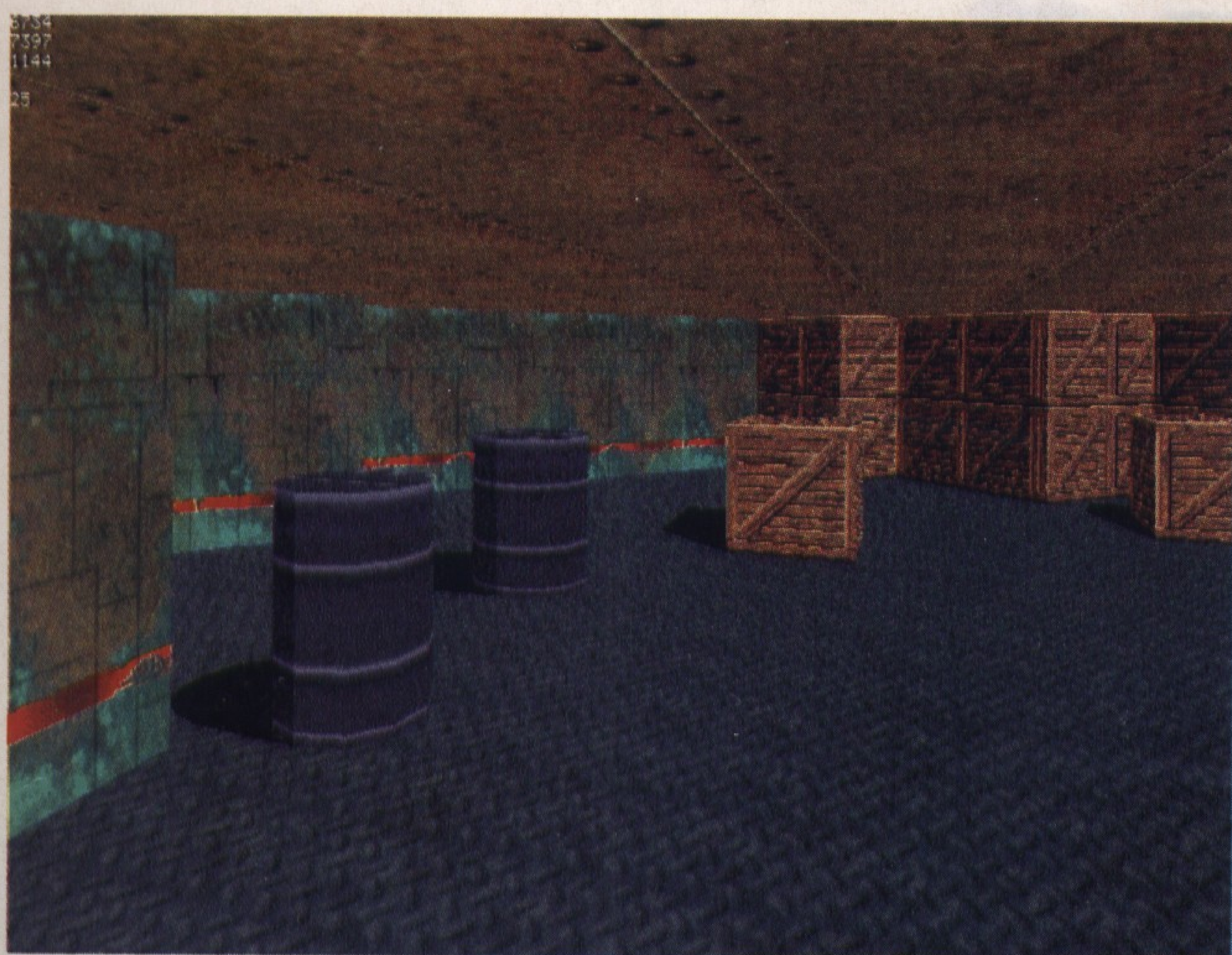
Para el mundo 3D hay un editor de mapas. Con él podremos poner todos los elementos que componen la arquitectura del nivel. También

Fallos corregidos en la segunda versión.

Existen una serie de fallos que se han arreglado de la versión anterior, hemos creado una lista con los más reseñables. No están todos los que son, pero sí son todos los que están. Ahí va la lista:

- Al hacer un shell y cambiar de unidad después no compilaba. SOLUCIONADO ! ! !
- Error al llamar a la función *put_sprite()* de una DLL. SOLUCIONADO ! ! !
- Errores de compilación de div.h SOLUCIONADO ! ! !
- Fallo *blastem'up file* por *fichero*. SOLUCIONADO ! ! !
- Error al introducir en un get un número fuera de rango y pulsar TAB. SOLUCIONADO ! ! !
- Fallo al calcular el offset de una estructura private o local (*offset estructura*), no sumaba el id.
- Arreglado fallo de private struct x[<constante>].
- Corregido error al arrastrar un mapa a un menú en segundo plano.
- Las casillas *Marcar/arrastrar* e *Info* de los FPG funcionan como globales, en lugar de locales. SOLUCIONADO ! ! !
- Se machaca el FPG si se *guarda como* con el mismo nombre. SOLUCIONADO ! ! !
- No se leen los BMP de 256 colores con opción de compresión activada. SOLUCIONADO ! ! !
- Fallo en la mascara del reborde de los font generados con iluminación. SOLUCIONADO ! ! !
- Al cargar un gráfico de FPG arrastrando, no mantiene su código y descripción. SOLUCIONADO ! ! !
- Al cargar un gráfico de FPG por menú, no se muestra el título correctamente. SOLUCIONADO ! ! !
- Ya no se permite meter en un FPG un mapa con un código mayor a 999 (!)
- Al cargar p.ej: KK.PCX de un FPG se puede *Guardar...*, sin tener el path!!! SOLUCIONADO ! ! !
- Al sustituir un graf. en un FPG (meter, borrar o cambiar código) se muestra una barra de progreso
- Problemas al borrar gráficos del los FPG (en ocasiones se queda el _DIV_.FPG) SOLUCIONADO ! ! !
- El generador de explosiones parece colgarse al generarlas grandes (ej. 1024x768) SOLUCIONADO ! ! !
- Fallo en los FPG al *Borrar marcados*, se abría un fichero que no se cerraba (y no se eliminaban los mapas) SOLUCIONADO ! ! !
- Al pasar a un modo de vídeo menor a 640x400, cambiar al font de editor de 6x8 (si se está utilizando uno mayor a 8x8).
- Error que provocaba que, al cambiar la paleta, a veces no se preguntara por la adaptación de todos los FPG. SOLUCIONADO ! ! !
- Al hacer un shell o *Guardar como* un sonido, dejaba de escucharse los sonidos en DIV (bajo W95) SOLUCIONADO ! ! !

Como podéis comprobar, algunos de ellos podían causar problemas bastantes graves. Gracias a todos por hacernos llegar vuestras sugerencias y haber reseñado estos fallos y otros que no están en la lista, pero que seguro están solucionados.



Los elementos se incorporan perfectamente a los mapeados.



Te sorprenderán las posibilidades de los juegos de luces.

se podrán poner banderas que indiquen puntos calientes dentro del mapa. Además se podrá elegir el dibujo que va impreso en cada una de las paredes, suelos y techos de nuestro mundo. Se pueden tener varios mundos creados dentro del escritorio y siempre es posible hacer cambios dentro de nuestro mundo, mientras nos acordemos de grabar el trabajo hecho.

Dentro del lenguaje hay rutinas que permiten controlar el mundo. Además de añadir procesos, del mismo que se hacía en los modos 7, disponiendo éstos de sus variables locales pertinentes. Se puede modificar la textura del mundo, así como mover suelos, techos y paredes. Siempre teniendo precaución en no provocar que algún elemento se salga de nuestro mundo. Con el editor de niveles y las nuevas rutinas de modo 8, se podrán hacer juegos en 3D de lo que se quiera, por ejemplo uno de fútbol de naves o de cualquier otra temática.

Más lenguaje

No sólo ha habido mejoras en el lenguaje con el tema de los modos 8. Además han incorporado rutinas que tratan otros temas, pero veamos qué temas han solucionado.

El primero de ellos es el de la búsqueda de caminos. En muchos juegos, como son los de aventuras gráficas o los de estrategia, algún elemento del juego tiene que ir de un punto a otro de un mapa, sorteando obstáculos por el camino. Dentro de DIV, este tema está solucionado, ya que se han creado una serie de rutinas en las que proporcionándole dos puntos, es capaz de devolvernos la ruta más óptima entre ellos, sorteando todos los obstáculos. La forma de utilizar estas rutinas es muy sencilla y no entraña ninguna complicación. Además existe un cuadro de diálogo para crear los mapas de búsquedas, facilitando así su creación.

Otro tema solucionado es el de la conexión entre ordenadores. DIV 2 incoporará rutinas de red que permitirán que dos o más ordenadores jueguen entre sí. Se podrán conectar dos ordenadores mediante un cable, red local o incluso mediante módem. Los programas no estarán tan solos: podrán competir con otros ordenadores.

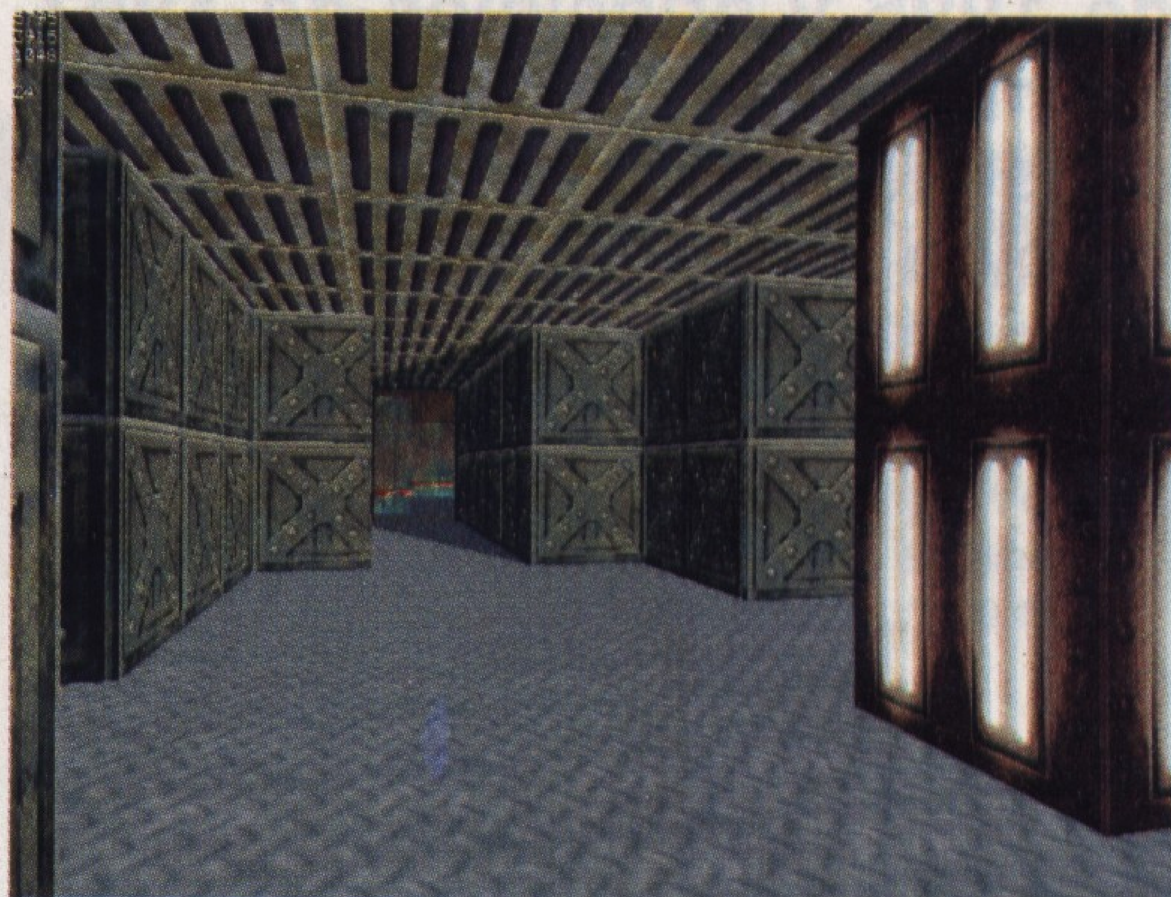
Una rutina permite crear mapas gráficos en memoria. Esto no parece útil, pero nos deja calcular y construir un gráfico desde cero.

Hay unas rutinas que se echaban en falta. En primer lugar están las que se encargan de textos, pero en formato ASCII. Es decir, hay rutinas que nos permitirán manejar cadenas de textos, pudiendo leerlas, escribirlas, cortarlas, pegarlas, etc. Además, se incluirán rutinas para que el usuario pueda introducir texto. Estas rutinas se han conocido siempre en el mercado informático como INPUT. Por ejemplo, son muy útiles a la hora de que el usuario introduzca su nombre en la tabla de records, apareciendo un cuadro de diálogo que lo permita.

Otras rutinas que se modifican son las de *load()* y *save()*. Aunque existían unas antes, se ampliarán. Ahora se podrá grabar en varias sesiones. Con esto se podrá tener en un solo fichero la información necesaria. También se grabarán y cargarán textos. Las operaciones con el disco duro quedan completadas.

Hay nuevas operaciones para trabajar con la paleta. Ahora se podrá definir cualquier color en ella. Así se definirá una paleta en tiempo de ejecución, cambiando los valores con programación.

En el apartado sonoro, se podrán cargar sonidos WAV desde el lenguaje. Existe la posibilidad de cambiar la frecuencia de sonido de los sampleados. Gracias a esto se conseguirán muestras de sonido de mayor calidad. Se quiere incluir otra rutina de música CD, pero depende del tiempo disponible.



La complejidad de los escenarios es enorme.

Sin salirnos del apartado sonoro, se incluyen rutinas para ejecutar ficheros S3M, que se pueden construir con gran cantidad de programas del mercado. Se ha creado una variable global llamada *fps*, que contiene el número aproximado de las imágenes mostradas por segundo. Así sabré si el juego va rápido o no. En el tema variables, ahora también se podrán crear tablas y estructuras de hasta 3 dimensiones, con el formato tabla [x,y,z]. Esto facilitará la programación y hará que el código del programa sea mucho más legible.

Otro de los detalles que salta a la vista es el de los nuevos listados coloreados

Otra función útil será *write_in_map()*, que permitirá escribir textos en los mapas gráficos. Se podrán tratar los textos como gráficos, pudiendo hacerlos rotar, escalar, etc., brindando nuevas expectativas a las presentaciones.

Por último, hay un nuevo DIV32RUN.DLL más compacto y sólido, pues este fichero se debe integrar en las instalaciones. Los juegos completos ocuparán menos.

Antonio Marchal

StRAtOS

Un grupo cuya única razón de ser son los videojuegos

StRAtOS es una asociación de desarrolladores de juegos y multimedia fundada por Antonio Arteaga y José Carlos García entre finales de 1996 y principios de 1997. En muy poco tiempo se ha consolidado como la asociación de este tipo más importante en España e Iberoamérica.

En sus orígenes, la asociación contaba con siete u ocho grupos miembros, tenía una reducida web de no más de tres páginas y los primeros juegos que publicó (las dos *Leyendas de Lhodrye*) ni siquiera mostraban el logotipo.

StRAtOS es una evolución lógica de un planteamiento básico. Al principio era un grupo de desarrollo que realizó alguna incursión en publicaciones de juegos. Debido a esta actividad se realizaron contactos con otros grupos de desarrollo de España y esto supuso el germen de la asociación.

Había varios grupos con características y problemas propios: falta de grafistas, programadores, compositores, etc. y ninguna idea acerca de cómo publicar sus juegos.

En la raíz de todo, la sensación de que los grupos principiantes y sin recursos estaban a merced de las grandes compañías. Había que buscar una forma de ayudar en lo posible. Y eso se conseguiría mediante la fuerza del número.

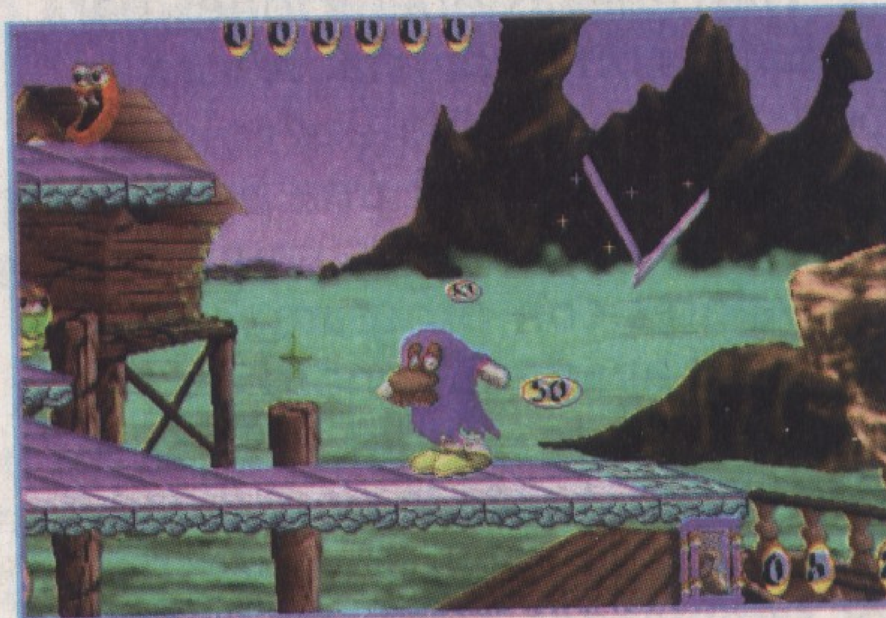
Miembros

Los principios fueron difíciles. Una asociación desconocida buscaba personas y/o grupos relacionados con el mundo del desarrollo, y lo hacía en los concursos de las editoriales, con anuncios, organizando concursos propios, a través de IRC.

Debido a la experiencia negativa de varios grupos frente a las distribuidoras, existía un enorme recelo. Según Antonio Arteaga: "si yo hiciese esto por dinero, hace tiempo que lo hubiese dejado todo. Al principio era casi un hobby, pero hoy se ha convertido en algo por lo que creo que merece la pena trabajar todos los días y a todas horas. El agradecimiento sincero de un grupo de chavales a los que has ayudado no se paga con dinero."

Poco a poco, el tiempo lo ha dejado claro. La asociación se ha volcado en apoyar a todos los grupos y personas que se han hecho miembros. Y los ocho grupos iniciales se han convertido en más de doscientos cincuenta, en España, Argentina, Cuba, Venezuela, Colombia, Uruguay, etc.

Un plataformas de buena factura.



Métodos de trabajo

Lo más complicada es la coordinación del potencial humano. Sería prácticamente imposible plantearse un método único de trabajo para los proyectos de todos los grupos. Cada cual tiene su técnica y sus líneas de desarrollo. Desde la asociación se orienta sobre los posibles fallos en los productos.

"Creo que una de las posibilidades más interesantes que ofrece StRAtOS, dejando a un lado la publicación de los títulos, es sin duda poder contactar con aquellas personas que necesitamos y que, sin saberlo, tal vez vivan a poca distancia de nosotros", opina A. Arteaga. "La lista de programadores, grafistas, compositores, guionistas, *testers*, etc. se está haciendo tan amplia que tenemos cubierto prácticamente el 80% del territorio español."

Compañías como Ubi Soft, Hammer Technologies o Pyro hacen llegar a la asociación sus necesidades de personal y desde ella se intenta contactar con aquéllos que pueden reunir los requisitos. Varios son los miembros de StRAtOS que actualmente están trabajando para dichas empresas.

Publicación de juegos y otros productos

Tercer objetivo: publicar los juegos de los miembros. Han salido a la luz más de catorce títulos, que han servido para promocionar a los grupos desarrolladores, además de suponer-

"Quedadas"

Para afrontar la edición 98 del SIMO, StRAtOS está organizando un encuentro entre todos los miembros y simpatizantes de la asociación. Muchos de los miembros se conocen desde hace tiempo, a través de teléfono, E-mail o IRC, pero no se han visto. Es una forma de acentuar el carácter humano de StRAtOS. En cualquier caso, aprovechando distintos eventos como la *Euskal Party*, varios miembros organizan sus propias "quedadas" para conocerse.

les el correspondiente beneficio económico y darles nuevas energías.

Prácticamente la totalidad de las publicaciones se han hecho a través de editoriales que han regalado los juegos junto a una revista. Los grupos no tienen un presupuesto fuerte y no pueden pasar todo el tiempo intentando desarrollar un juego de la calidad comercial más alta.

Sin embargo, el acabado de los juegos es cada día mejor. Sin abandonar la línea editorial, StRAtOS mantiene en la actualidad conversaciones con empresas nacionales e internacionales para conseguir la comercialización de los títulos.

"No está muy lejos el día en que podrán adquirirse juegos de StRAtOS en las grandes superficies, en los kioscos de prensa o en las webs de los grandes distribuidores", asegura Antonio Arteaga. "Si todo funciona correctamente, podría decir que el año que viene habremos alcanzado otro de los sueños de la asociación: la publicación de la práctica totalidad de los productos que reúnan unos mínimos requisitos de calidad."

Medios de apoyo

"No podemos olvidar todos los recursos que nos han ayudado a llegar donde estamos", continúa Arteaga, "Unos nos los hemos tenido que trabajar, otros nos han venido solos a medida que la asociación ha ido creciendo." Por recursos entenderemos los distintos medios que han servido de apoyo a StRAtOS para darse a conocer. Detallémoslos:

Página web en *http://www.stratos-ad.com*

En la actualidad el sitio está formado por cerca de 130 páginas, con secciones de noticias, contacto, información, ofertas de trabajo, etc. La asociación cuenta con dos revistas electrónicas: GargonScene, que tras un periodo de descanso se reanudará con el número 5, y StRActOS, alojada en la web, que comprende artículos técnicos, opiniones, experiencias personales, consejos y guías, etc.

Antonio Arteaga.





Perspectiva caballera que intenta emular a los grandes del género.

"No cabe duda de que Internet ha supuesto una herramienta fundamental en el desarrollo y mantenimiento de la asociación. No podría imaginarme de qué forma moverlo todo sin el correo electrónico, de cómo llegar a todos y cada uno de los miembros al mismo tiempo y con efectividad. Todavía hay algunos grupos y personas que carecen de conexión a la red, pero con el tiempo esto se solventará", dice Arteaga.

Canal #programacion_stratos en el IRC hispano

Sirve de punto de encuentro habitual de muchos miembros de StRAtOS. También hay muchas personas que visitan de manera asidua el canal simplemente porque en él hay "buen rollo".

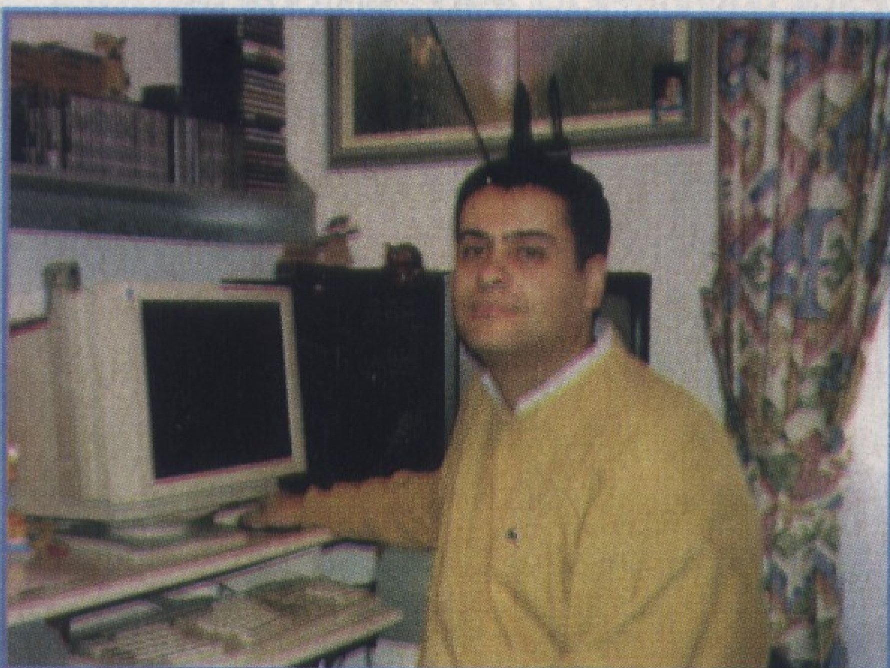
Apariciones en los medios de comunicación

Cuando StRAtOS comenzó su andadura, prácticamente suplicó a alguna revista que les dedicase un hueco en uno de los números. A medida que la asociación ha tomado cuerpo y, los reportajes o entrevistas en revistas o programas especializados de radio han aumentado.

Proyectos en desarrollo

Con tal cantidad de grupos, existe mucha producción. En estos momentos hay cerca de una treintena de títulos en marcha. Los lenguajes y herramientas son de lo más variado, al igual que los entornos para los que se destinan. Se están creando aventuras, juegos de estrategia, arcades, plataformas, juegos de habilidad, simuladores, deportivos. Existe una tendencia a programar para Windows y a sacar el máximo provecho de las posibilidades del nuevo hardware.

José Carlos García.





Una perspectiva casi cenital para un programa de estrategia.

El Canal #programacion_stratos en el IRC hispano sirve de punto de encuentro habitual de muchos miembros de StRAtOS. También hay muchas personas que visitan de manera asidua el canal simplemente porque en él hay "buen rollo".

Según Arteaga, "van quedando atrás los juegos donde se cuenta la historia en un texto, se entra directamente a la acción y se termina con una pantalla de felicitación. Eso estaba bien para la sobrecarga de títulos shareware que viajaban y viajan por todo el mundo con la pretensión de vender unas pocas unidades a bajo precio. Si queremos ser realistas y hacernos un hueco en este mundo entrando con la cabeza alta, tenemos que tender a lo que quiere y se merece el mercado final. Ya sabemos que no podemos competir con las grandes compañías en estos momentos, porque obviamente (de momento) los recursos son insuficientes para abordar esa batalla, pero al menos podremos decir que nuestros juegos son divertidos y están hechos con buen gusto."

El hecho de que algunas casas de software de otros países acudan a StRAtOS para que les oriente o consiga distribución para sus productos es un síntoma de que la asociación es más fuerte de lo que cabría suponer.

Redacción Div Manía



TokenKai: estrategia pura

Aparece el primer programa profesional realizado con DIV

El primer juego que demostrará que DIV Games Studio es una herramienta mediante la cual es factible realizar programas que se encuentren en la primera línea del mercado estará entre nosotros en las próximas Navidades. Su nombre es TokenKai y se centra en el apasionante mundo de la estrategia en tiempo real.



Hoy por hoy, la programación de un videojuego depende de un equipo compuesto por un buen número de hombres. La imagen del loco programador de Spectrum que se las veía y se las deseaba consigo mismo durante horas ha desaparecido hace ya bastante tiempo. Y, si las motivaciones de una sola persona son ya de por sí un tanto complejas, ¿cuánto lo serán las de un grupo de hombres? En efecto, ¿qué será lo que mueva a tanta gente para la consecución de un mismo objetivo?

Todo los amantes de la estrategia pueden preparar sus neuronas para enfrentarse a un excelente programa que va a poner a prueba toda su capacidad de acción y de manejo de tropas

Probablemente cada uno de los grafistas, cada uno de los programadores, de los guionistas y, en definitiva, de todos los que participan en el proyecto tiene sus motivaciones particulares que en nada se asemejarán a las de su vecino más inmediato. Pero, por encima de las razones particulares, hay algún motivo que los empuja en una misma dirección, de eso no hay duda. Las más de las veces, seguramente no será otra cosa que el dinero. Pero en otras ocasiones, afortunadamente cada vez más numerosas, lo que los mueve es el deseo de hacer algo de calidad, algo grande, algo que repercuta en los otros grupos de hombres que hacen el mismo trabajo que ellos. Y que, sólo entonces y lógicamente, dé dinero.

Así pues, cabe preguntarnos qué está moviendo a los chicos de Hammer Technologies a realizar el programa que presentamos aquí, *TokenKai*. Porque un programa de esta envergadura requiere una inversión enorme de tiempo, de energía, de horas de sueño y también, nuevamente, de dinero. Pues bien, por encima de todo este programa nació del deseo de demostrar que era posible realizar un producto que alcanzase una calidad de primera línea mediante la herramienta de programación DIV, que tanto éxito ha tenido entre los usuarios.

DIV corría el peligro de convertirse en una herramienta secundaria, no apta para la realización de los grandes programas. O, más exactamente, empezaba a respirarse en el ambiente que era un medio limitado para producir un juego. La segunda edición de DIV corrige las limitaciones que podían haber llevado a semejante conclusión y, para corroborarlo definitivamente, el equipo de Hammer Technologies se puso a trabajar en un título que se basase en dicha herramienta. Esperamos que a partir de ahora sean muchos más los que veamos aparecer en el mercado.

La elección de los programadores es todo un reto. Han decidido centrarse en uno de los géneros que se encuentran en auge en estos momentos. Y esto tiene sus ventajas y sus desventajas. Por una parte tiene un público asegurado que probablemente se acercará al título sin preguntarse con demasiada energía qué tiene de original, qué ofrece en lo que respecta a la calidad. Es el público incondicional de un género que levanta pasiones en estos



Enfréntate a una primera misión que no te va a dejar ni respirar un momento.



escenarios aparecen sometas.



Al protagonista, como es lógico, lo van a moler a palos: podrás ver el estado del mismo en la parte derecha.



Los detalles *gore* no son demasiados, pero lo que sí es seguro es que podrás ver bastante sangre.

momentos, el público que, por qué no decirlo, se traga todo lo que le echen sin demasiados miramientos.

Pero a quienes hay que convencer es a los otros, a los que de verdad consiguen que un programa se convierta en un auténtico *crack*, los que se lo piensan dos veces y, además, han asimilado las innovaciones, los más importantes avances del género. Y aquí es donde se entra en un terreno difícil. El listón del género en estos momentos está muy alto, altísimo, y es necesario realizar un esfuerzo ímprobo para alcanzar el nivel de calidad mínimo necesario para empezar a pensar en lo más grande.

Echemos un vistazo ahora al proceso que ha seguido este grupo de programación español. Siempre es algo apasionante descubrir el proceso creativo de un programa. En el proyecto ha intervenido un total de diez personas, entre programadores, diseñadores 3D, infografistas, guionistas y músicos.

El software empleado para la realización de los escenarios ha sido principalmente 3D Studio MAX, aunque también se han utilizado otras aplicaciones como Character Studio, módulos y *plug-ins* de partículas dinámicas, generado de render por ray-tracing y Photoshop. Los ordenadores personales en los que se utilizaron dichos programas eran principalmente PCs con placas duales Pentium 300 bajo Windows NT y 98.

Para realizar todo el modulado inorgánico (es decir, escenarios y objetos) y el orgánico (personajes) han partido de bocetos realizados a mano por dibujantes expertos. Estos dibujos eran escaneados y se utilizaban como calco en las plantillas de 3D Studio MAX. Hay que señalar en este sentido, por último, que todo el desarrollo del programa se ha hecho en un plazo aproximado de ocho meses.

DIV facilitó en gran medida la labor de programación, sin que eso sirviese de detrimento para la calidad final del producto. La agilidad que la nueva herramienta presta a quien la utiliza acorta el tiempo que es necesario invertir para lograr un producto de estas características. Obviamente no se tiene la absoluta libertad de quien prefiere revisar hasta el más mínimo detalle todos sus



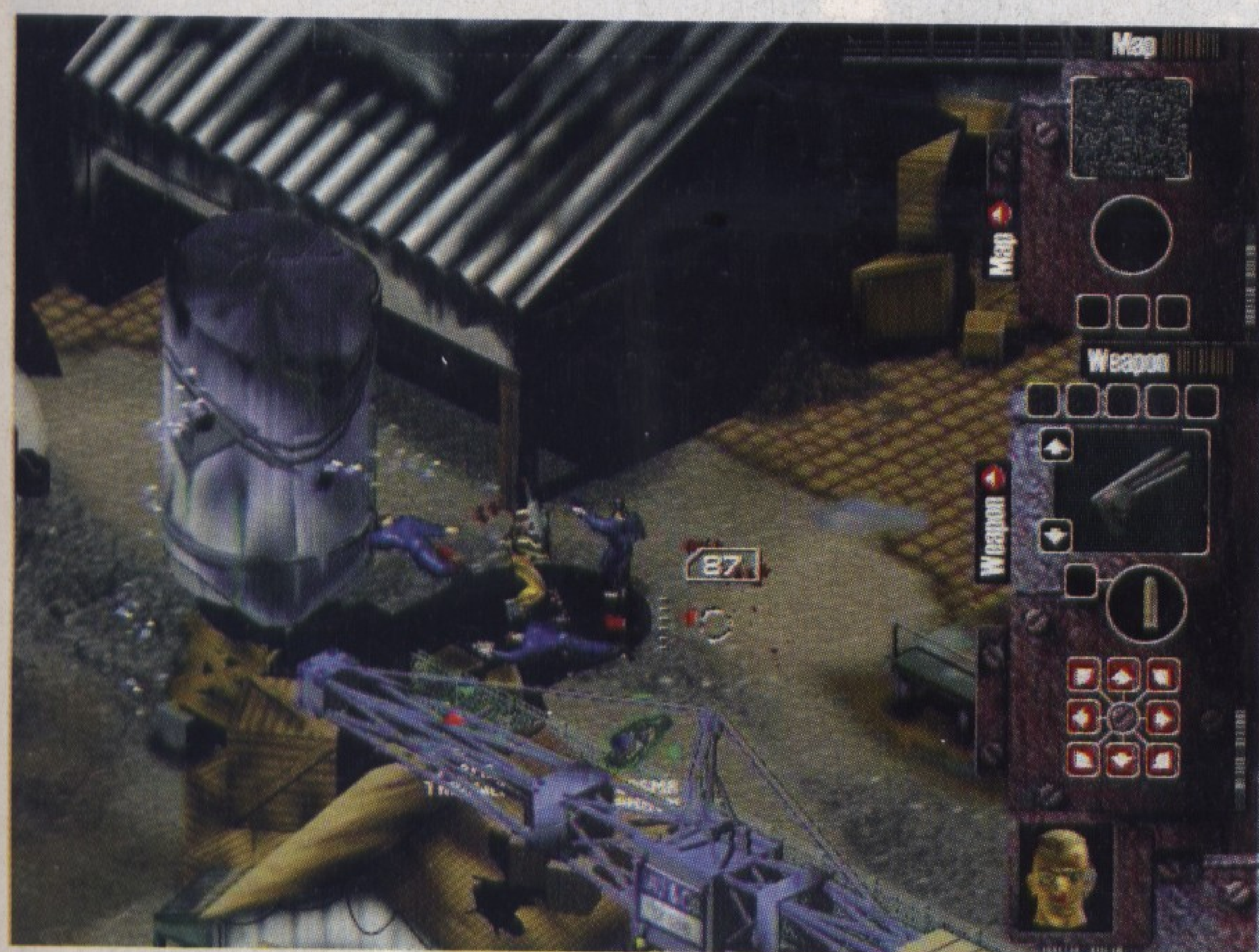
A medida que vayas recibiendo golpes el personaje principal se irá desfigurando poco a poco hasta morir.



Los tiroteos en los que te ves envuelto se ven fielmente reflejados en las distintas formas de proyectil.



En los elementos que cubren los escenarios aparecen todos los destrozos a los que los sometas.



Al protagonista, como es lógico, lo van a moler a palos: podrás ver el estado del mismo en la parte derecha.



Los detalles *gore* no son demasiados, pero lo que sí es seguro es que podrás ver bastante sangre.

momentos, el público que, por qué no decirlo, se traga todo lo que le echen sin demasiados miramientos.

Pero a quienes hay que convencer es a los otros, a los que de verdad consiguen que un programa se convierta en un auténtico *crack*, los que se lo piensan dos veces y, además, han asimilado las innovaciones, los más importantes avances del género. Y aquí es donde se entra en un terreno difícil. El listón del género en estos momentos está muy alto, altísimo, y es necesario realizar un esfuerzo ímprobo para alcanzar el nivel de calidad mínimo necesario para empezar a pensar en lo más grande.

Echemos un vistazo ahora al proceso que ha seguido este grupo de programación español. Siempre es algo apasionante descubrir el proceso creativo de un programa. En el proyecto ha intervenido un total de diez personas, entre programadores, diseñadores 3D, infografistas, guionistas y músicos.

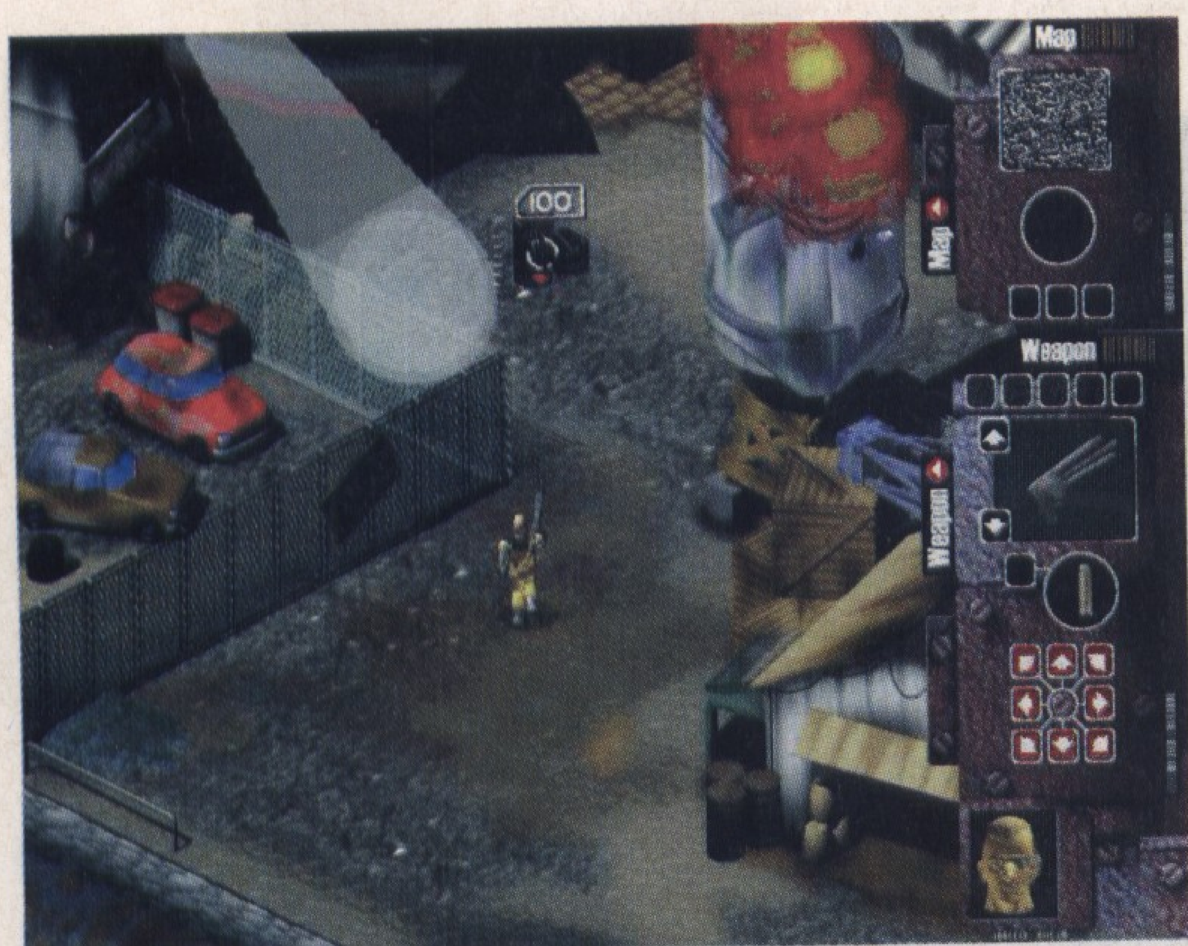
El software empleado para la realización de los escenarios ha sido principalmente 3D Studio MAX, aunque también se han utilizado otras aplicaciones como Character Studio, módulos y *plug-ins* de partículas dinámicas, generado de render por ray-tracing y Photoshop. Los ordenadores personales en los que se utilizaron dichos programas eran principalmente PCs con placas duales Pentium 300 bajo Windows NT y 98.

Para realizar todo el modulado inorgánico (es decir, escenarios y objetos) y el orgánico (personajes) han partido de bocetos realizados a mano por dibujantes expertos. Estos dibujos eran escaneados y se utilizaban como calco en las plantillas de 3D Studio MAX. Hay que señalar en este sentido, por último, que todo el desarrollo del programa se ha hecho en un plazo aproximado de ocho meses.

DIV facilitó en gran medida la labor de programación, sin que eso sirviese de detrimento para la calidad final del producto. La agilidad que la nueva herramienta presta a quien la utiliza acorta el tiempo que es necesario invertir para lograr un producto de estas características. Obviamente no se tiene la absoluta libertad de quien prefiere revisar hasta el más mínimo detalle todos sus



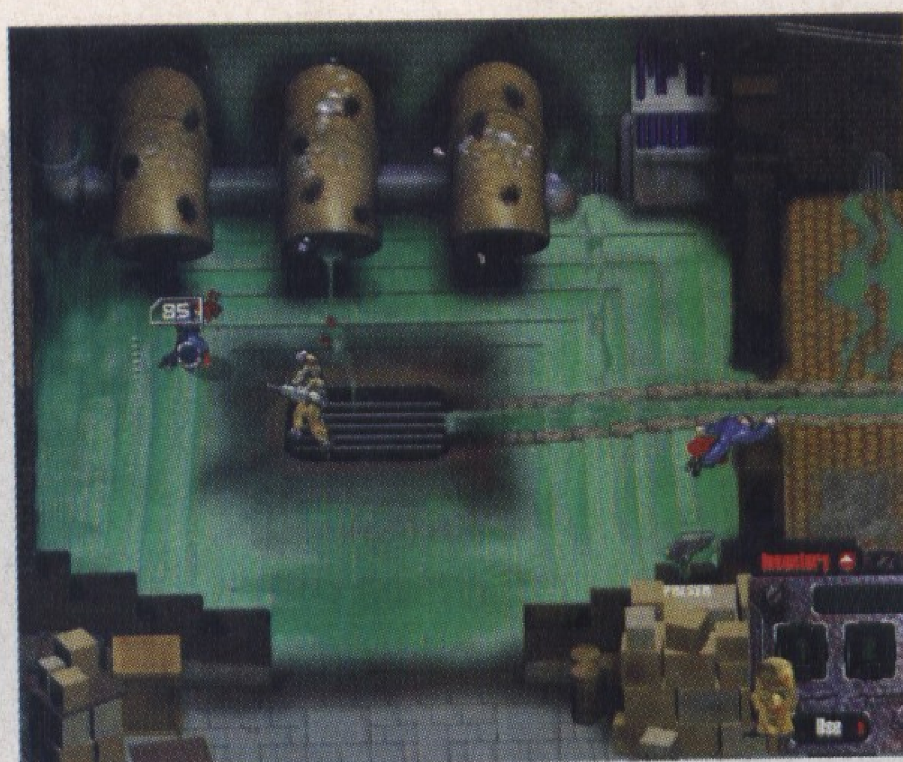
A medida que vayas recibiendo golpes el personaje principal se irá desfigurando poco a poco hasta morir.



Mantente alejado de las explosiones, que no no van a dejar títere con cabeza.



Incluso en los interiores del programa debes estar alerta: aquí vemos un garaje sangriento.











códigos. Sin embargo, DIV 2 ha sido optimizado y no ofrece problemas incluso a los programadores más exigentes.

En el entorno, realizado con la ya típica visión isométrica de los escenarios en que se desarrolla la acción, han preferido obviar los elementos poligonales que, por ejemplo, nos ofrecía *Wargames*, en beneficio de la rapidez y la jugabilidad del programa. Nos recuerda más a títulos reconocidos como *Syndicate* que a las últimas novedades.

La vista caballera que domina durante todo el desarrollo del juego nos permitirá escapar con mayor rapidez de los peligros que nos acechan y asestar golpes mortales a los enemigos

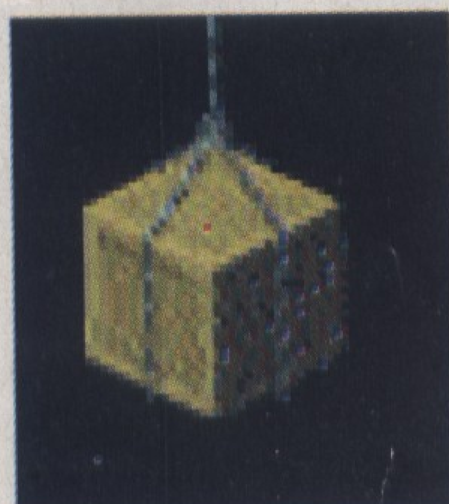
El trabajo realizado con los mapeados ha sido realmente exhaustivo. La extensión de los mismos es enorme, de modo que satisfará sin problemas a todos aquellos que deseen perderse en la inmensidad del ordenador. Tampoco hará falta ir dejando miguitas de pan, pero al menos es posible que la acción se desarrolle de forma óptima.

Los personajes del juego y las animaciones que lo amenizan han sido muy cuidados en todos los sentidos. De hecho, el programa incluye más de 10.000 fotogramas que se encargan de realizar las distintas animaciones. Esto incluye, naturalmente, algunos de los efectos *gore* que no faltan en las últimas producciones: restos sangre en las paredes, hilillos de lo mismo en labios semiabiertos, y un largo etcétera.

También los sonidos han sido cuidados hasta el más mínimo detalle. Oirás desde los mismos roces de los proyectiles contra las paredes hasta casquillos saliendo disparados del arma. El aspecto sonoro, hoy en día, cobra cada vez más importancia; no se puede olvidar que los efectos gráficos cobran mucha más energía si están bien acompañados por sonidos.

Por otro lado, los responsables del juego nunca olvidaron que, si se trataba de un programa de estrategia, el argumento era muy importante. A pesar de que los ingredientes arcade son tantos que prácticamente igualan los factores estratégicos, gracias a Dios no han dejado de lado la parte argumental que muchas veces abandonan los productores.

Así, han decido ambientar el juego en un futuro muy cercano, apenas treinta años después de que haya empezado el nuevo milenio. La violencia se ha adueñado de nuestro planeta de tal forma que casi no hay otra posibilidad de vivir que relacionarse con ella de una manera u otra. Las altas esferas de la sociedad son sólo corrupción, y todo aquello que tenga que ver con el poder y con el dinero se solventa de mala manera. Es decir, por medio de grupos mercenarios que sacan a todos las castañas del fuego... O hacen arder las castañas de los demás.








Muchos de los elementos que aparecen en el juego tienen una participación activa en el mismo. Por ello, deberás estar atento a todo, incluso a las inocentes cajas que penden por encima de tu cabeza.



HAMMER TECHNOLOGIES

En los últimos años, el software patrio está viviendo un auge como nunca vimos antes. Un buen ejemplo de ello es la compañía Hammer Technologies, que en muy poco tiempo ha adquirido una fuerza poco común en un medio siempre difícil. Hay que señalar, de entrada, que esta compañía es la que ha puesto en marcha la herramienta de programación DIV, que ha motivado la aparición de esta revista y en breve verá su segunda versión, mejorada y actualizada.

El primer juego que ha aparecido bajo los auspicios de este sello es *Snowboard Avalanche*, un estupendo simulador de *snowboard* que puso muy alto el listón de calidad del software español. Continuó con un programa de tenis de gran calidad, que supo hacerse un hueco en un género tradicionalmente dominado por productos extranjeros. Su nombre era *Tie Break Tennis*, y proporcionaba altas cotas de calidad gráfica, así como de jugabilidad.

Mención especial merecen los programas que todavía mantiene en preparación la compañía y que aparecerán, probablemente, a principios del



año que viene. El primero de ellos es *Jagdverband 44*, *Jag 44* para los amigos. Con el escalofriante subtítulo de *Screaming Eagles*, se trata de un simulador de vuelo que nos lleva hasta los tiempos de la Segunda Guerra Mundial. Han tomado personajes y equipos reales; de hecho, el que da nombre al programa, muy al estilo de *Red Baron*, era un conocido escuadrón de la Luftwaffe en el que estaban los mejores pilotos del ejército alemán. Para la realización de las misiones también han decido tomar referentes reales, teniendo en cuenta desde las campañas que hubo hasta los escenarios en que se desarrollaron.

Tienes la posibilidad de tomar cualquiera de los dos bandos que se encontraban en lucha. Los aliados se ven representados por la octava fuerza aérea (8th Air Force), y los alemanes por la mencionada Luftwaffe. Los aparatos que se encuentran a tu disposición también han sido fielmente reproducidos a partir de los de la época. Mencionaremos sólo el B-17 Flying Fortress y el BF-110 de visión nocturna.

El comportamiento de cada uno de los aviones será más o menos diferente según sus características; sin embargo, no habrá que realizar un costoso estudio de los manuales, pues se ha preferido, en general, que primen los factores arcade de la simulación. No hay que entender por ello que han descuidado la física del programa; al contrario, la tecnología empleada intenta reflejar al máximo la realidad. Los aficionados a los simuladores de vuelo de época no podrán perderse ente título.

Otro de los programas en el que la compañía ha cifrado sus esperanzas para el futuro más inmediato es *Neon Angel*. Se trata de la primera incursión en el campo de las plataformas 3D que inauguró el genial *Tomb Raider*. Es un terreno peliagudo y quizá el proceso de desarrollo de este programa será más largo que el promedio habitual de un videojuego. Se está realizando una exhaustiva labor en la elaboración de caracteres con una personalidad marcada. Las imágenes que acompañan esta nota dan una completa idea de cómo se está trabajando en este sentido.

Los escenarios tridimensionales en que se desarrolla la acción son el otro punto fuerte del programa. En un título de este género se trata de uno de los detalles más importantes, que no puede ser descuidado. Se está cuidando la realización de los mapeados con mimo, así como los juegos de luces y sombras y las texturas escogidas. El programa está ambientado en un mundo futurista en el que los clones humanos se han convertido en algo cotidiano.

Un último comentario para la segunda versión de DIV, que también corre a cargo de esta compañía. La nueva herramienta saldrá a la calle en breve plazo. Entre sus características más destacadas se encuentran las rutinas de caminos, especialmente aptas para programas de estrategia, librerías de red para la realización de programas multiusuario y un *engine* 3D que lo hace apto para arcades 3D.



Tokenkai es uno de esos grupos mercenarios que tienen que hacer el trabajo sucio. De hecho, es el mejor de ellos, a menos que tu impericia acabe con su fama. Tú te encargarás de dirigirlo. Cualquiera que tenga el dinero suficiente se convertirá en el patrón de tu grupo: desde gobiernos hasta las más diversas mafias, pasando por grandes empresas o poderosos acaudalados.

Las misiones que se encomienden el equipo pueden ser de lo más variado. No olvides que se trata de un grupo de combate elite, y como tal es capaz de llevar a buen término las misiones más suicidas que te puedas imaginar. El grupo es capaz de realizar su trabajo en cualquier terreno, de modo que a lo largo del programa visitarás los más extraños lugares del planeta.

Y cuando decimos extraños, no nos referimos a la muralla China, las Montañas de la Luna o el Machu Pichu, sino a lugares como alcantarillas, aeropuertos, subterráneos, parques de atracciones, etc. En principio, habrá un total de quince misiones en las que deberás emplear toda tu habilidad para sobrevivir y conseguir los objetivos marcados.

Rafael María Claudín

Introducción al diseño gráfico

Un primer acercamiento a las imágenes

A partir de éste y durante los próximos artículos, mostraremos lo básico del diseño, y se trabajará con los programas 2D y 3D que hay en el mercado. Incluyendo el editor gráfico de DIV Games Studio.

Ahora se explicarán algunas cosas que aunque para muchos de vosotros sean elementales, hay que incidir en ellas ya que constituyen la base de un buen diseño, ya sea para un juego o para un programa.

Ejemplo de sumario con su filete vertical al lado
Ejemplo de sumario con su filete vertical al lado
Ejemplo de sumario con su filete vertical al lado

El tamaño de la imagen es un factor a tener en cuenta. Por ejemplo: resultaría un poco absurdo realizar el dibujo de un coche a un tamaño de 320x200 pixeles si después, en la pantalla, sólo se verá en 60x60 pixeles. Lo que pasaría si hiciéramos esto sería:

- La imagen ocuparía memoria innecesariamente.
- El juego o programa iría más lento.

La imagen se vería mal por culpa de intentar reducir muchos pixeles en pocos.

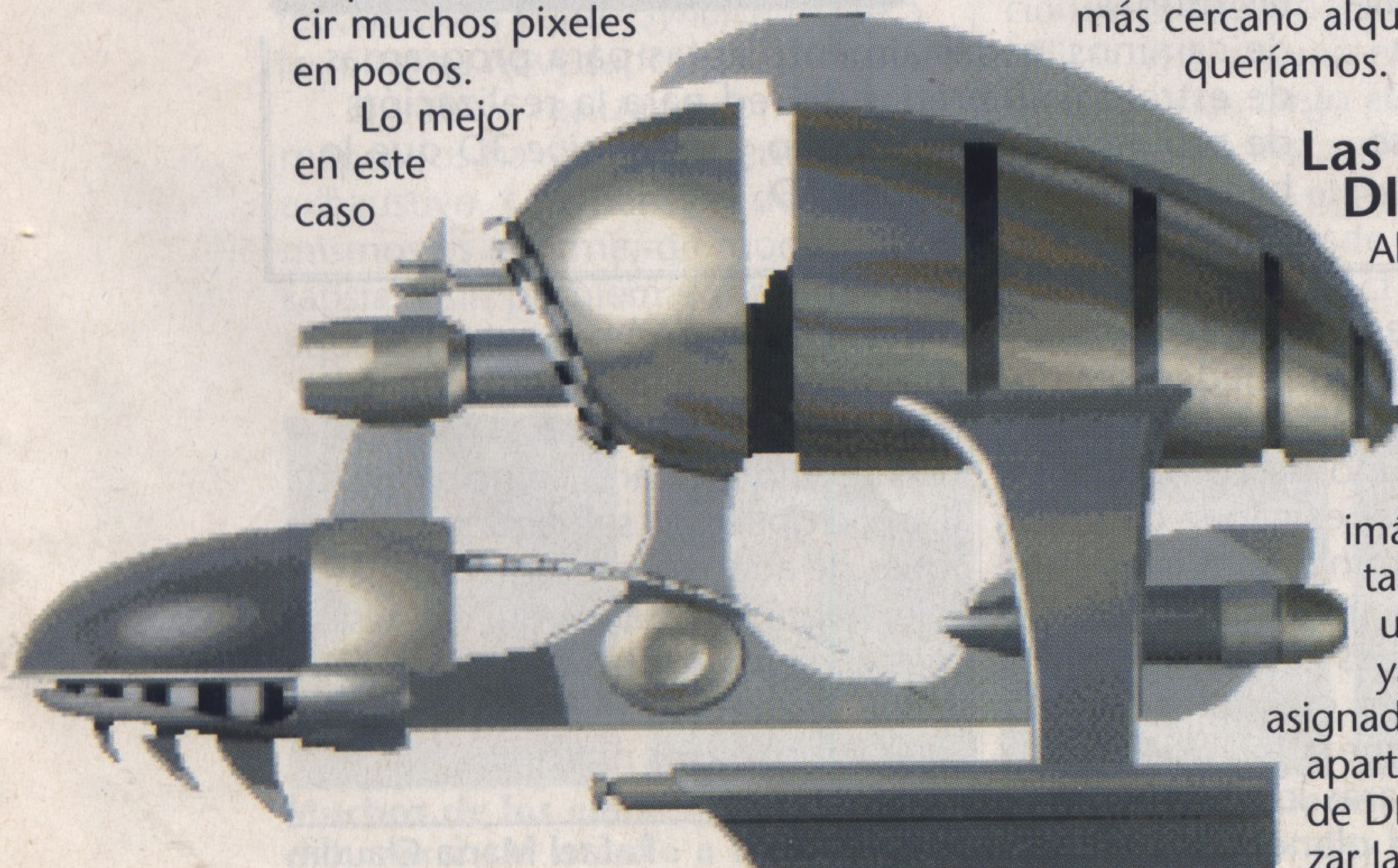
Lo mejor en este caso sería intentar simplificar el gráfico lo máximo posible, sin hacer muchos detalles, y hacerlo directamente con su tamaño original.

Los colores y paletas para usar en DIV también son temas que se deben tener en cuenta. Aunque la paleta de 256 colores de DIV parece un poco limitada, se pueden conseguir resultados muy profesionales y bien acabados.

A la hora de generar una imagen, deberemos jugar mucho con los colores y pensar antes de empezar:

- Qué elementos habrá en la pantalla, con qué colores y qué cantidad de tonos. Si por ejemplo, queremos que en la pantalla aparezca un dibujo de una selva con un cielo azul degradado, la imagen que generemos debería tener en la paleta bastantes verdes y gran cantidad de azules. De esta manera, podemos hacer una selva, que al pasarla a 256 colores no nos estropee el dibujo.

La creación de paletas personalizadas es muy útil cuando se trata de utilizar imágenes con gran cantidad de colores, ya que crea una paleta con los colores que hay actualmente en la imagen. Es obvio que si la imagen tiene más de 256 colores, al pasarla a esta resolución no nos quede igual, pero se conseguirá el resultado más cercano alque nosotros queríamos.



Las paletas en DIV

Al cargar el fichero para gráficos de cualquier programa, y presentar una de sus imágenes en pantalla, saldrá con una paleta que ya se le habrá asignado. Es decir, aparte de la paleta de DIV puedes utilizar las que hay en el subdirectorio PAL del directorio de DIV. Otra opción es crearte tu nueva paleta personalizada, con la que podrás aprovechar mucho mejor la imagen y sus colores.

- Se va a presentar un ejemplo con Photoshop. Partimos de la Imagen 1, a 8 bits de color y la queremos pasar a 256 colores para utilizarla con DIV, se hace de la siguiente manera:
- Se va a *Modo* y se pincha sobre *Color Indexado* (256 colores).

Después se despliega el menú de *Paleta* y se le dice *Adaptable*.



IMAGEN 1. Ejemplo del degradado a 8 bits de color.

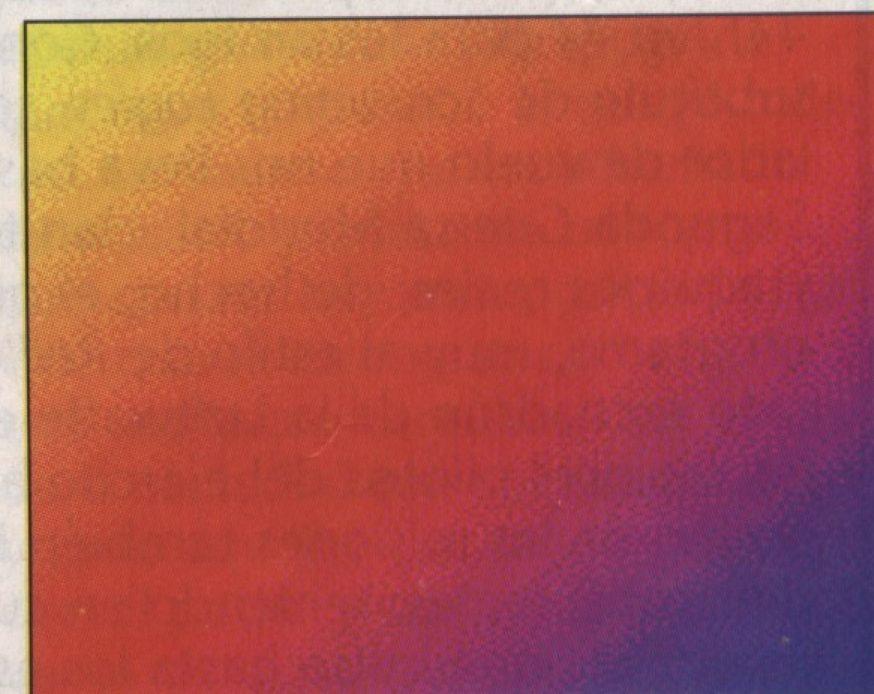


IMAGEN 2. Ejemplo del degradado a 256 *colores paleta Windows*.

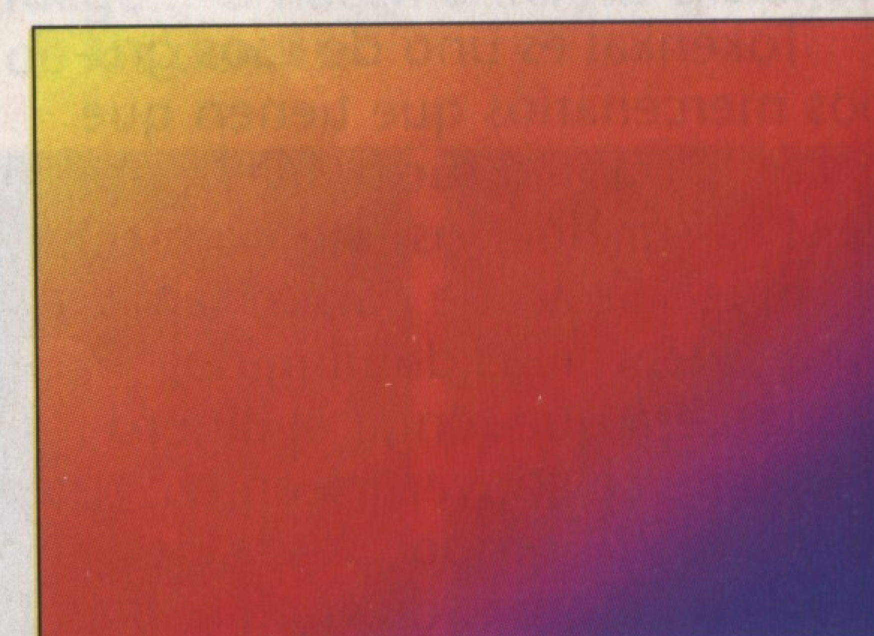


IMAGEN 3. Ejemplo del degradado a 256 colores *paleta Adaptable*.

De esta manera, crearemos una paleta que aprovecha al máximo los colores de nuestra imagen. Para visualizar dos imágenes simultáneamente en la pantalla del programa de DIV, debemos usar la misma paleta para las dos imágenes. De lo contrario, surgirían problemas. Si lo que queremos es mostrar otra imagen con otra paleta en otra parte del juego donde no salga la primera imagen, sólo debemos volver a cambiar la paleta desde DIV (además, deberemos usar otro FPG).

Para asignar la misma paleta a más de una imagen, lo que deberemos hacer es pasar la imagen a color indexado y decirle *A Medida*. Nos saldrá una lista desplegable con las paletas a elegir. Entonces, se le debe decir *Anterior*.

Seguidamente, para ver nuestra efectividad con los cambios de paleta, se probará con DIV. Se debe crear un FPG (fichero para gráficos) en el que se introducirá la imagen con la que hemos jugado con las paletas. Al cargar la imagen, DIV nos mencionará si queremos adaptar la paleta de la imagen y le decimos que no. Dependerá de si decimos sí o no, para que DIV y toda la paleta del juego sea la de esa imagen. En ciertos casos se deberá decir que sí. Estos casos se explicarán más adelante en otros artículos.

Nota: se pueden utilizar diferentes FPG con diferentes paletas.

El programa es para todos

Una cosa muy importante es tener en cuenta el ordenador que tiene el usuario o destinatario de nuestro programa/juego. Es decir, siempre que se vayan a poner grandes resoluciones o gráficos impresionantes, debemos pensar en si el ordenador que recibirá estos gráficos, puede soportar tales resoluciones o colores.

Actualmente, casi todo el mundo tiene un ordenador con una tarjeta de vídeo que, como mínimo, soporta los 256 colores y resoluciones de al menos 640x480. Eso significa que no podemos hacer un programa que sólo funcione con los 1024x768 pixeles de pantalla y que además tenga 16.000 colores. Aunque de momento DIV sólo soporta los 256 colores, se estima que en futuras versiones aceptará más colores. Así pues, es importante el hecho de que aunque el programa con el que trabajamos acepte grandes resoluciones, siempre debemos pensar que hay alguien con un ordenador no tan potente como el nuestro.

Breve resumen de las paletas

- Deben tenerse en cuenta los colores de la imagen para hacerse la paleta.
- Al guardarla, se debe hacer recomendablemente con *paleta Adaptable*.
- Si otra imagen tiene la misma paleta que la imagen anterior, debe guardarse como *paleta Anterior*.

Nota: el cambio de paleta se puede hacer también desde otros programas, como Corel.

Un ejemplo práctico

Se van a generar unas letras con una sombra para un título que saldrá en la pantalla, al ejecutar un programa hecho con DIV (también hecho en esta sección).

Primero de todo se debe ir a Photoshop, donde crearemos una imagen de unos 320x200 pixeles y crearemos unas letras que digan *Gráficos*. Seguidamente pintaremos estas letras de negro y les aplicaremos el filtro de *Desenfoque Gaussiano*. De esta manera, tendremos unas letras difuminadas como las de la Imagen 4. A continuación, volvemos a crear un texto, esta vez de otro color o de negro, pero sin aplicarle el filtro. Al hacer esto, tendremos un resultado como el de la Imagen 5.

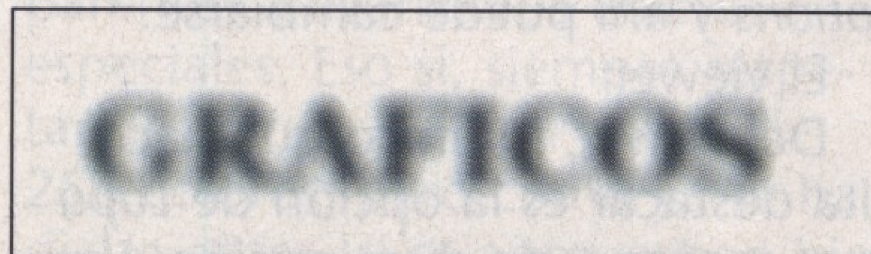


IMAGEN 4. Ejemplo de desenfoque.



IMAGEN 5. Acabado de la imagen.

Bien, ya tenemos la imagen, por lo que ya podemos pasarla a color indexado. Los pasos ya comentados son: ir a *Modo/Color Indexado* y decirle en la elección de las paletas: *Adaptable*.

Seguidamente, se debe ir a DIV, crear un en el subdirectorio de FPG de DIV y después insertar nuestra imagen al FPG en el espacio número 1.

Cuando carguemos el fichero de la imagen de las letras con sombra, nos dirá que si queremos que DIV adapte esa paleta, responderemos que no.

Guardaremos este archivo como *grafi1.fpg*. Una vez hecho esto, pasaremos al código fuente en el que haremos que nos muestre esta imagen y que cuando pulsemos espacio, salgamos del programa.

A continuación, creamos un nuevo programa llamado *grafi1.prg* y copiamos en su contenido lo siguiente para que funcione:

```
PROGRAM grafi1;
BEGIN
load_fpg("grafi1.fpg");
put_screen(0,1);

LOOP
frame;
if(key(_space))
frame; break; frame;
end
END
frame;
END
```

Cuando se acabe de copiar esto, se ejecuta para ver si el resultado es el esperado.

El resultado vendría a ser algo como la Imagen 6. Si es así, ¡felicidades!



IMAGEN 6. Imagen total de lo que se ha hecho.

Antes de empezar con los programas de diseño 2D y 3D del mercado, es bueno practicar con las paletas, aunque sea sólo con barras de colores; cambiar la paleta en medio de un programa (explicado en el próximo artículo pero también en el manual de DIV); ir probando a hacer composiciones de diferentes imágenes antes de mostrarlo todo junto en la pantalla (todo con la misma paleta).

Hoy por hoy los gráficos son uno de los factores más importantes dentro de un videojuego, por lo que cualquier atención para ellos es poca.

Los visores más conocidos

Entre los visores de imágenes que existen para plataforma PC, se deben destacar dos. El primero de ellos es el Sea, que se distribuye Shareware con los CDs de algunas revistas. El segundo es el Acdsee que últimamente se ha hecho muy popular, este segundo programa para muchos, es el mejor, más rápido y fácil de utilizar del mercado.

Puede abrir gran cantidad de formatos, aunque no pueda con-

vertir a tantos como lee. Aparte de ser capaz de abrir los archivos convencionales, el Acdsee es capaz de abrir ficheros PIC de Softimage y convertirlos a JPG, BMP u otras extensiones.

El programa se divide en dos partes que son:

- El *browser* (buscador)
- *Viewer* (Visor)

En la Imagen 7 podemos observar el *browser* del Acdsee y en la Imagen 8 el *viewer* del mismo programa.

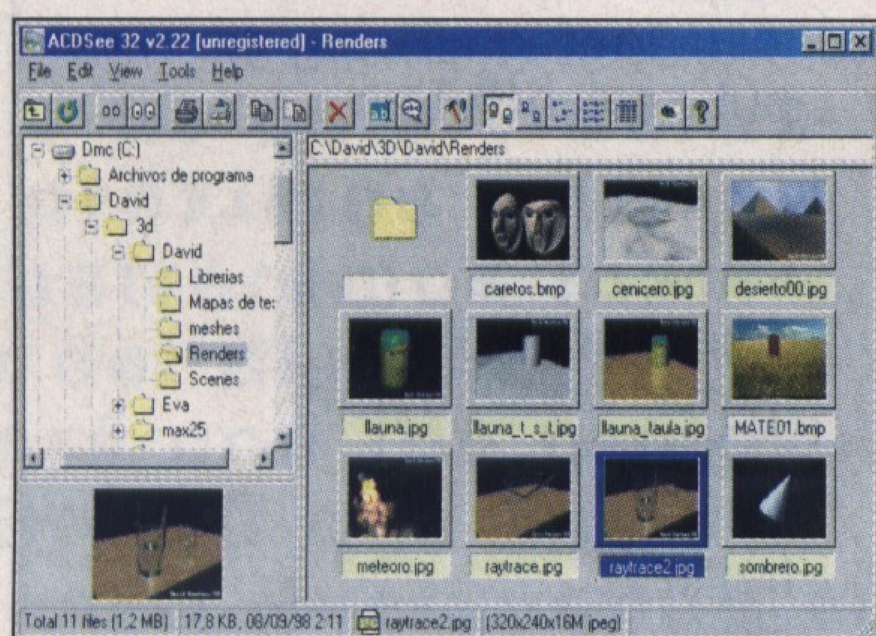


IMAGEN 7. Captura del *browser* del Acdsee.



IMAGEN 8. Captura del *viewer* del Acdsee.

Funcionamiento del Acdsee

Como ya se ha dicho anteriormente, el Acdsee es uno de los visores más fáciles de usar del mercado. A continuación se mostrarán sus opciones básicas, y se transformará algún archivo para ver su facilidad de uso y su excelente efectividad.

Adcsee es un excelente visor de imágenes. Su manejo es sencillo e intuitivo y cualquier usuario puede aprenderlo en un momento.

- El browser

En el menú *File* hay las opciones de: *New Folder, Open, Print Setup, Print, Exit* que son: *Nueva Carpeta, Abrir, Configurar Impresora, Imprimir* y *Salir* respectivamente.

En el menú *Edit* hay las opciones de: *Cut, Copy, Copy Image, Delete, Select All Files, Copy to, Move to, Rename, Describe* que son *Cortar, Copiar imagen, Borrar, Seleccionar todos los ficheros, Copiar a, Mover a, Renombrar* y *Describir* respectivamente.

Nota: la opción *Cut* es para hacer un *Cortar* y *Pegar* (no se selecciona un trozo de la imagen sino toda). Con *Copy* copiamos las imágenes seleccionadas, con *Copy Image* sólo una. Y finalmente, con *Describe* nos hacemos una descripción de la imagen.

En el menú de *View* (que no se describirá porque no es algo imprescindible). Sólo hay opciones para definir el *Orden* de los ficheros, la forma en que se presentan.

Finalmente, queda el menú *Tools*, uno de los más interesantes y útiles que tiene este programa.

En este menú hay las opciones de: *Slide show, Slide show recursive, Convert, Generate file listing, Set wallpaper, Shell, Options, Register* que son: *Paso automático de imágenes, Automático de imágenes* pero entrando en subdirectorios del *Actual, Convertir, Generar lista de archivos, Poner imagen como fondo pantalla, Ejecutar, Opciones* y *Registro* respectivamente.

Nota: Generate file listing proporciona un fichero con la lista de imágenes y su tamaño en pantalla, su tamaño en bits y su descripción.

Truco: para aumentar o disminuir la velocidad en que el *Slide show* va mostrando automáticamente en la pantalla. Puede irse a *Options* y ahí puede cambiarse.

- El viewer

Del *viewer*, lo único que hace falta destacar es la opción de *Lupa* de aumento y de disminución. Los otros menús, son iguales que los del *browser*.

Convirtiendo ficheros con Acdsee

Ahora se va a mostrar el modo en que se puede cambiar la extensión de una imagen usando Acdsee. Los pasos a seguir son:

- Abrir el *browser* del Acdsee.
- Buscar la imagen *planeta.bmp* en el CD-Rom.
- Copiarla a un directorio del disco duro.
- Pulsar sobre *Tools/Convert* o bien *Crl+F*
- Convertirla a *planeta.jpg*.
- Mirar si se ha creado.

Una vez seguidos estos pasos, ya dispondremos de una imagen JPG. Debe seguirse el mismo proceso para guardar la imagen en formato PCX.

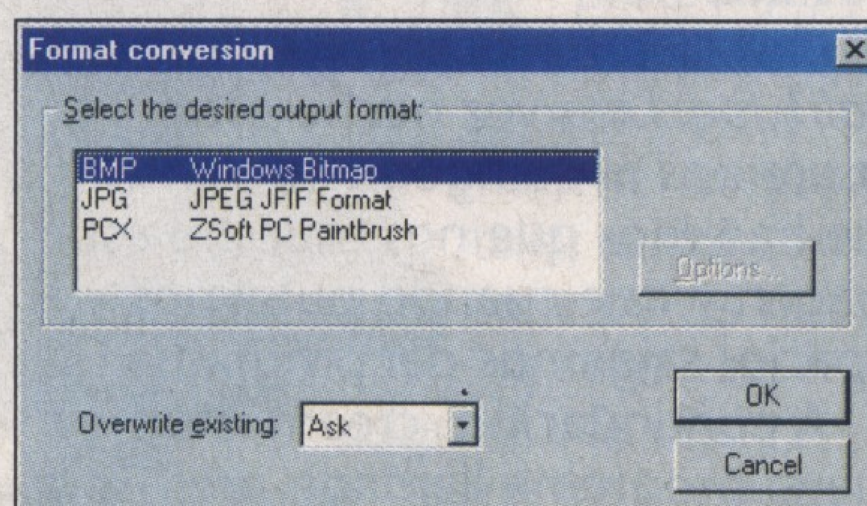


IMAGEN 9. La ventana de convertir.

Programas de Retoque fotográfico

Para desarrollar un buen diseño, también es bueno disponer de un programa de retoque fotográfico como el Adobe Photoshop (quizá uno de los más potentes y utilizados).

En los próximos capítulos, donde se utilice un programa de retoque fotográfico, posiblemente será el Photoshop, ya que es uno de los más extendidos, útiles e intuitivos.

En la Imagen 10 podemos observar la pantalla principal de Adobe Photoshop.

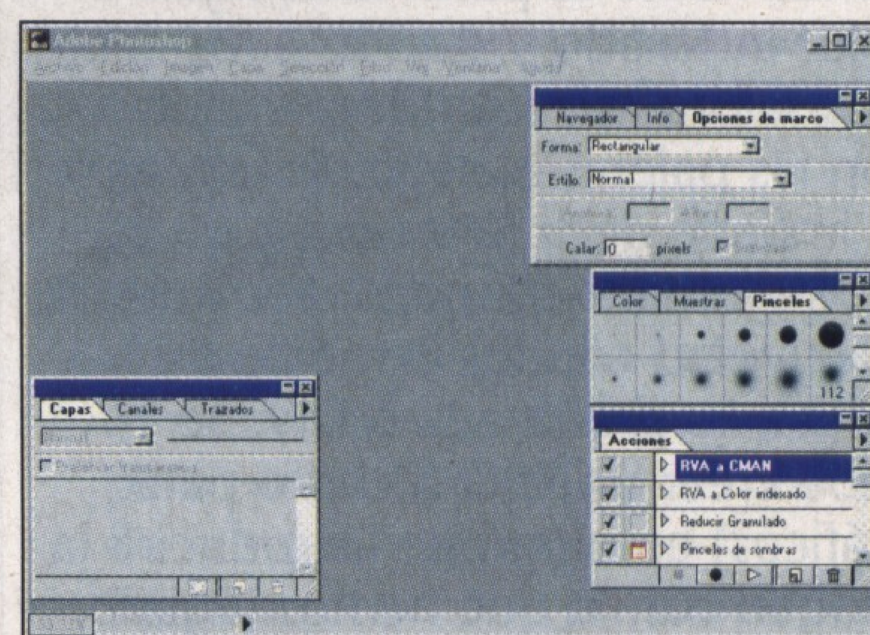


IMAGEN 10. Pantalla principal de Adobe Photoshop.

Funcionamiento de Photoshop

Una de las mayores ventajas de Adobe Photoshop es que es capaz de trabajar en capas, de manera que puedes poner una capa con una imagen y una opacidad del 100% y otra capa con otra imagen y con una visibilidad del 27 %. Aparte de las capas, también tiene una gran cantidad de útiles Plug-Ins como el ya comentado antes *Desenfoque Gaussiano.* También debe destacarse la cantidad de tipos de archivo capaces de abrir. Su compatibilidad con *Rva, Cman, Color Indexado, Escala de grises,* entre otros, hacen de Photoshop un programa excelente.

Los programas de 3D

En muchos videojuegos, se ven muchas veces gráficos en 3D, pero hay que distinguir entre dos tipos de 3D.

- Sólo gráficos en 3D (juego 2D pero gráficos 3D).
- Totalmente 3D (personajes 3D, entorno 3D).

En nuestro caso, sólo nos centraremos en el primero, ya que la versión actual de DIV no soporta 3D, pero sí que soporta cualquier tipo de gráfico.

Los 3 ejes de coordenadas.

Existen 3 ejes de coordenadas que son: «X»,«Y»,«Z» -*Ancho, Alto* y *Profundo* respectivamente. En la Imagen 11 pueden apreciarse los 3 ejes de coordenadas.

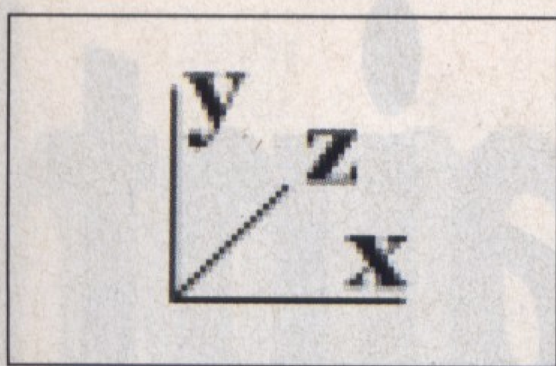


IMAGEN 11. Ejemplo de ejes de coordenadas.

Los mejores programas existentes para PC de 3D son:

- 3DS MAX.
- Softimage.
- Lightwave.
- Rhiino 3D (reciente aparición).

No los abarcaremos todos ni mucho menos, si nos centramos en alguno, será en el 3DS MAX y en el Rhino 3D (modelador en nurbs muy potente).



IMAGEN 12. Logotipo del MAX 2.

Con 3DS MAX se han hecho los gráficos de diversos juegos, entre ellos, el *Comandos* de Pyro Studios (buen juego, y no porque sea español). En el *Comandos*, se demuestra la potencia de 3DS MAX combinada con la originalidad y la habilidad de modeladores y animadores 3D.

El Rhino 3D también es un programa muy potente de reciente aparición. Su gran potencia para modelar con nurbs, su simplicidad de uso y su capacidad a exportar sus objetos para abrirlos desde programas como MAX, hacen que sea uno de los mejores por no decir el mejor modelador de Nurbs que actualmente hay en el mercado.

De momento, el Rhino aún está en versiones Beta y es un programa Shareware que también dan con las revistas y que no tiene ninguna o casi ninguna limitación.

El aspecto de Rhino es el de la Imagen 12.

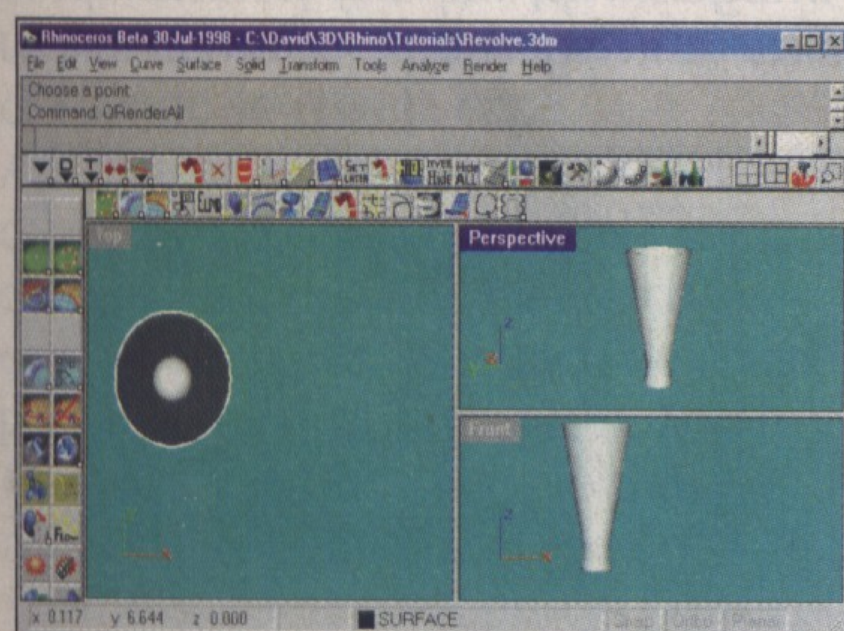
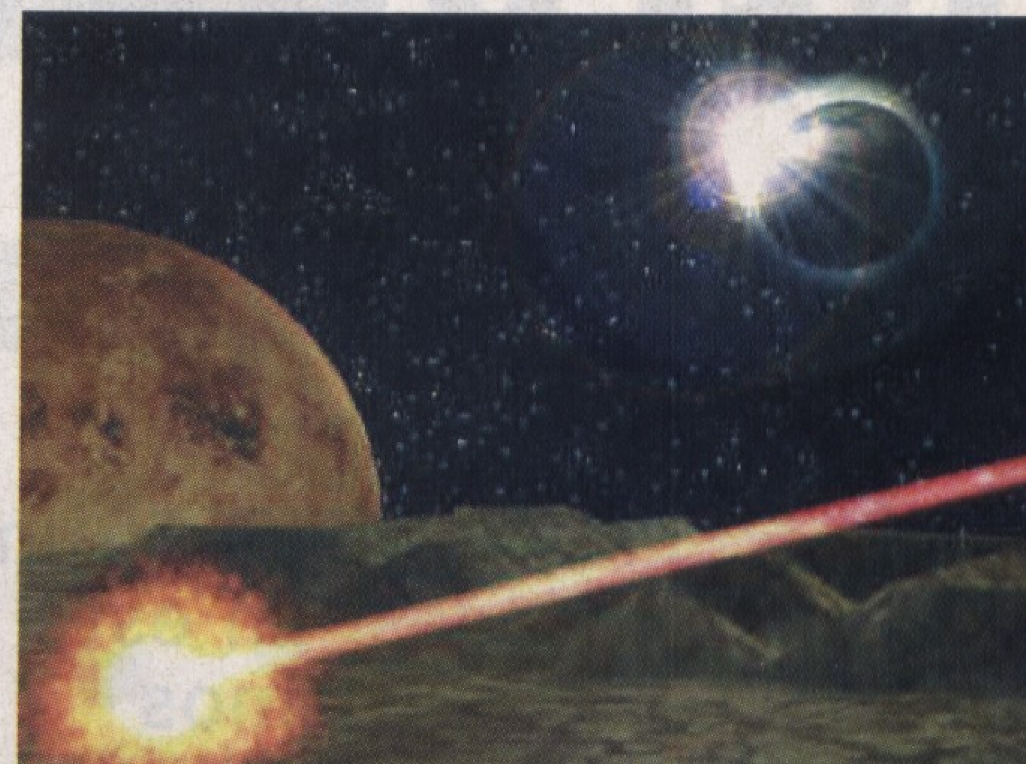


IMAGEN 12. Aspecto del programa Rhinoceros 3D.

El gran esquema

- Lo básico del diseño (tamaño, colores...).
- Las paletas de DIV (sólo 256 colores de momento).
- Guardar como *paleta Adaptable.*
- Si es repetida la paleta *A Medida* y *Anterior.*
- Tener en cuenta quién va a utilizar nuestro programa.
- Qué se usará más en estos artículos.
- Los visores más conocidos.
- Acdsee.
- Programas de Retoque fotográfico.
- Photoshop.
- Programas de 3D.
- 3DS MAX.
- Rhinoceros 3D.



El editor gráfico de DIV Games Studio

Finalmente, como herramienta que también usaremos en esta serie de artículos sobre diseño y dibujo, usaremos el editor gráfico de DIV. Una de sus grandes ventajas es el botón de *Deshacer* que va borrando cada cosa que has hecho, indefinidamente o casi (yo nunca he llegado a acabarlo). Aparte de esa ventaja, tiene muchas más, desde colores transparentes a avanzados efectos especiales. Eso sí, siempre respetando sólo hasta una paleta de 256 colores, pero como ya se ha dicho antes, si se aprovechan bien esos 256 colores, se pueden obtener maravillas de resultados. Al igual que todos los programas convencionales de dibujo, el editor gráfico de DIV lleva implementadas las herramientas de *Lápiz, Aerógrafo, Línea,* etcétera. Pero además, también lleva varias herramientas para hacer cosas como el espejado de una imagen (también se puede hacer mediante código), efectos luminosos...



En los próximos artículos en los que se tratará el editor de DIV, ya se darán por sabidas cosas que estén en el manual de DIV. De esta manera, se ganará tiempo y podremos pasar a trucos, técnicas, etcétera.

Si tienes alguna duda, no dudes en pasarte por el canal de DIV en el IRC *#DIV* donde podrás hacer tus preguntas y raramente no te las podrán contestar. O bien envíame un E-mail a *ksc_dmc@geocities.com* si crees que es una pregunta de interés general. En este caso, intentaré poner en algún hueco la respuesta.

En los próximos artículos ya empezaremos con el editor gráfico de DIV y haremos algún juego. Nos dedicaremos más a la practica. En este artículo había mucha teoría y quizás te hayas aburrido un poco, pero con los próximos artículos a ver si se anima la cosa. Hasta el próximo número. Un saludo.

David Martínez

Introducción a Paint

Los nuevos pinceles para la pantalla de tu PC

Paint Shop Pro 5 presenta un sinfín de novedades. En este artículo las expondremos brevemente: capas múltiples, numerosos niveles de anulaciones, tubos y sellos, herramientas de deformación libre, etc.; todo un mundo a un precio de 22.000 pesetas.

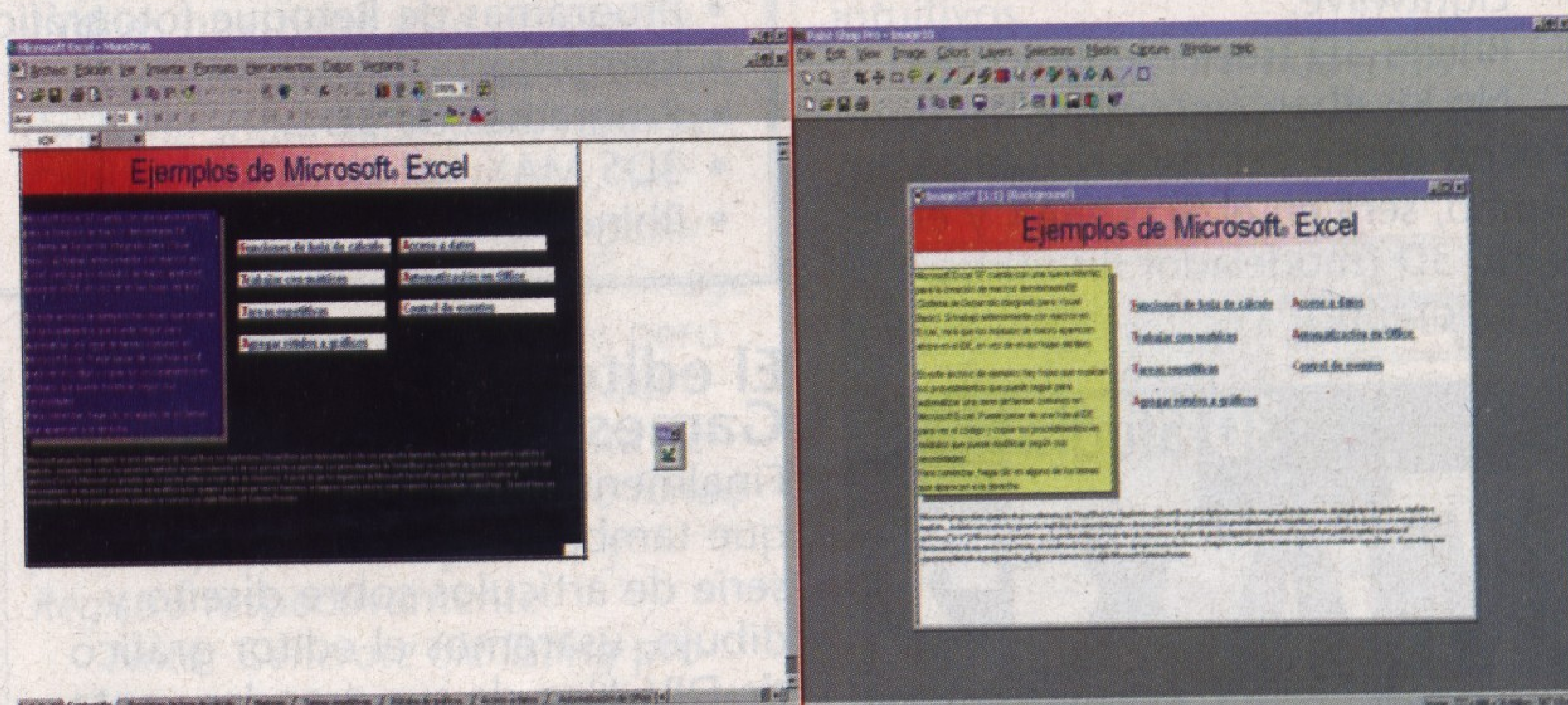


Figura 2. Así es la ventana que aparece cuando pinchamos en New.

New y *Open* son las primeras opciones. *New* despliega una ventana para especificar las características de la nueva imagen. Cada vez que se cambie un parámetro, se nos informa de la memoria que necesitaremos para crear una imagen así. Información que se muestra en el mismo grupo de las *Image Characteristics* con la etiqueta *Memory Required*.
Browse también se encuentra bajo *File*. Después están las opciones para importar y exportar, *Batch Conversion, Preferences* y, finalmente, *Exit*. En *Import* aparecen dos formatos: fuentes compatibles TWAIN y Kodak Digital Camera. *Batch Conversion* sirve para cambiar una gran cantidad de ficheros gráficos de formato.
El proceso comenzará cuando se pinche en el botón *Start*. Desde *Preferences* se controla la mayoría de parámetros configurables en Paint Shop Pro 5: localización de programas *plug-in*, formatos de ficheros, Monitor Gamma, etcétera.

Edit

En este grupo de menús aparece *Paste as new image*, que «transforma» contenidos del portapapeles en imágenes. Cuando no hay imágenes abiertas, *Edit* despliega una ventana con la herramienta comentada y otra: *Empty clipboard*, que eliminará cualquier dato guardado en el portapapeles.
Cuando hay una imagen abierta, *Edit* contiene herramientas como *Cut, Copy* y *Clear* y, de nuevo, *Paste*. El contenido del portapapeles puede ahora pegarse como nueva imagen, como capa de la imagen, como área de selección y como área de selección vacía.

View

View presenta las opciones de visualización: modo de pantalla, *zoom in* y *zoom out*, información sobre la imagen seleccionada, activar *grid*, mostrar reglas verticales y horizontales y configurar las barras herramientas. PSP 5 posee tres modos de pantalla: *Full Screen Edit, Full Screen Preview* y *Normal Viewing*.
Image Information abre una ventana en la que el programa desvela la información de la imagen. Además, permite introducir información mediante *Creator Information*.

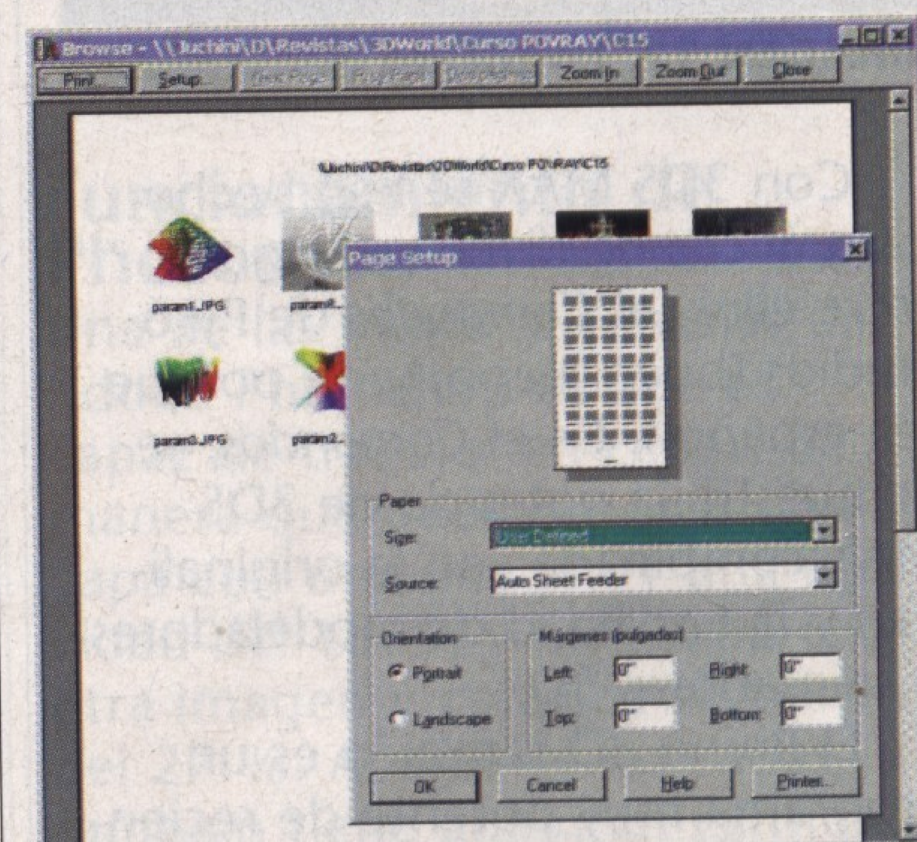


Figura 1. Aspecto del programa al ejecutarlo.

Imagen

El menú comienza con tres opciones para modificar la orientación de la imagen: *Flip, Mirror* y *Rotate*. *Flip* realiza la imagen espejo horizontal,

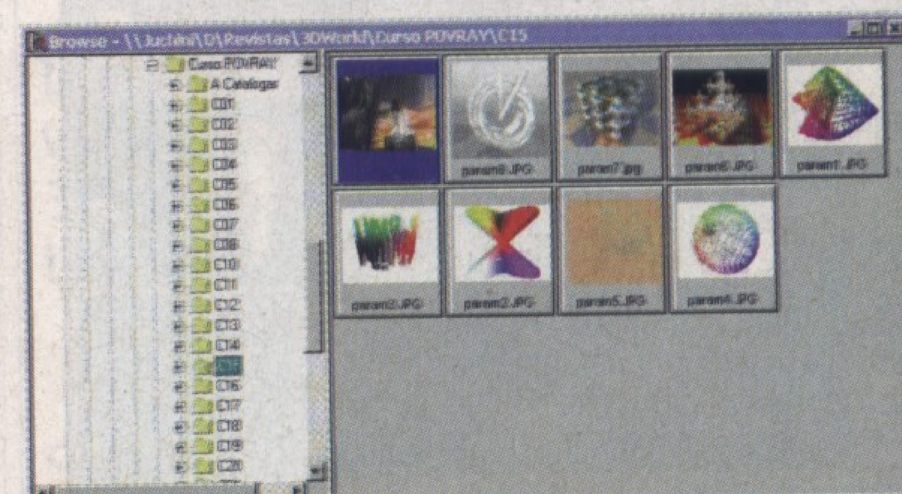
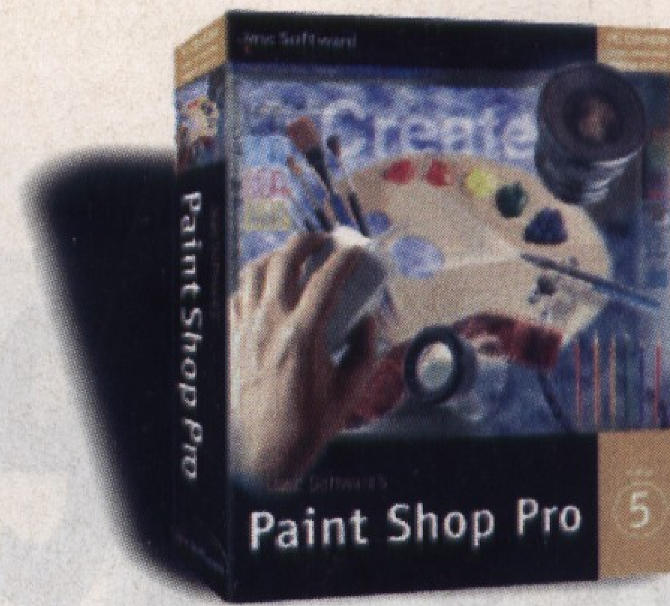


Figura 3. La ventana correspondiente al Browser del programa.

Resumen de las propiedades de Paint Shop Pro 5

- Multicapas
- Niveles de anulaciones múltiples
- Tubos / sellos
- Herramienta de deformación libre
- Pinceles para retocar
- Ojeador mejorado
- Separación CMYK
- Unidades de medida
- Tablas gráficas con sensibilidad a la presión del estilo
- Soporte de la norma ICM 2.0 para Win98/NT5
- Paleta de pinceles
- Pinceles personalizados
- Recuadro simplificado
- Herramientas de selección adicionales
- Compatibilidad con los plug-in PhotoShop
- Compatibilidad con los filtros de importación/exportación de Photoshop
- Soporte de formato de archivo PSD
- Conexión con cámaras fotográficas digitales
- Nuevo módulo de animación

Shop Pro 5



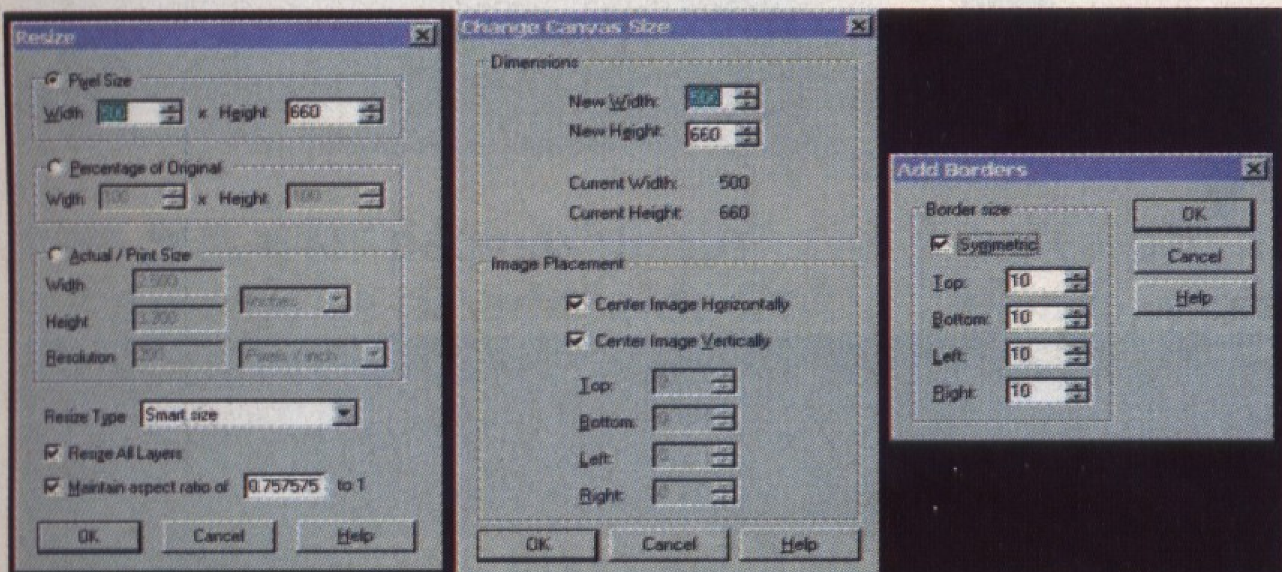


Figura 5. Las ventanas correspondientes a Add Border, Canvas Size y Resize del menú Image.

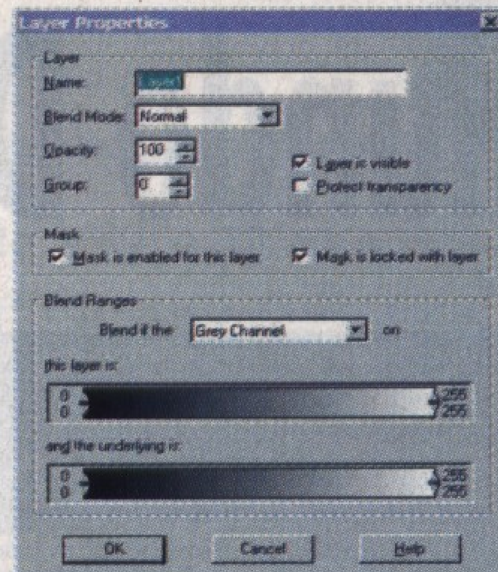


Figura 6. La ventana de New Layer.

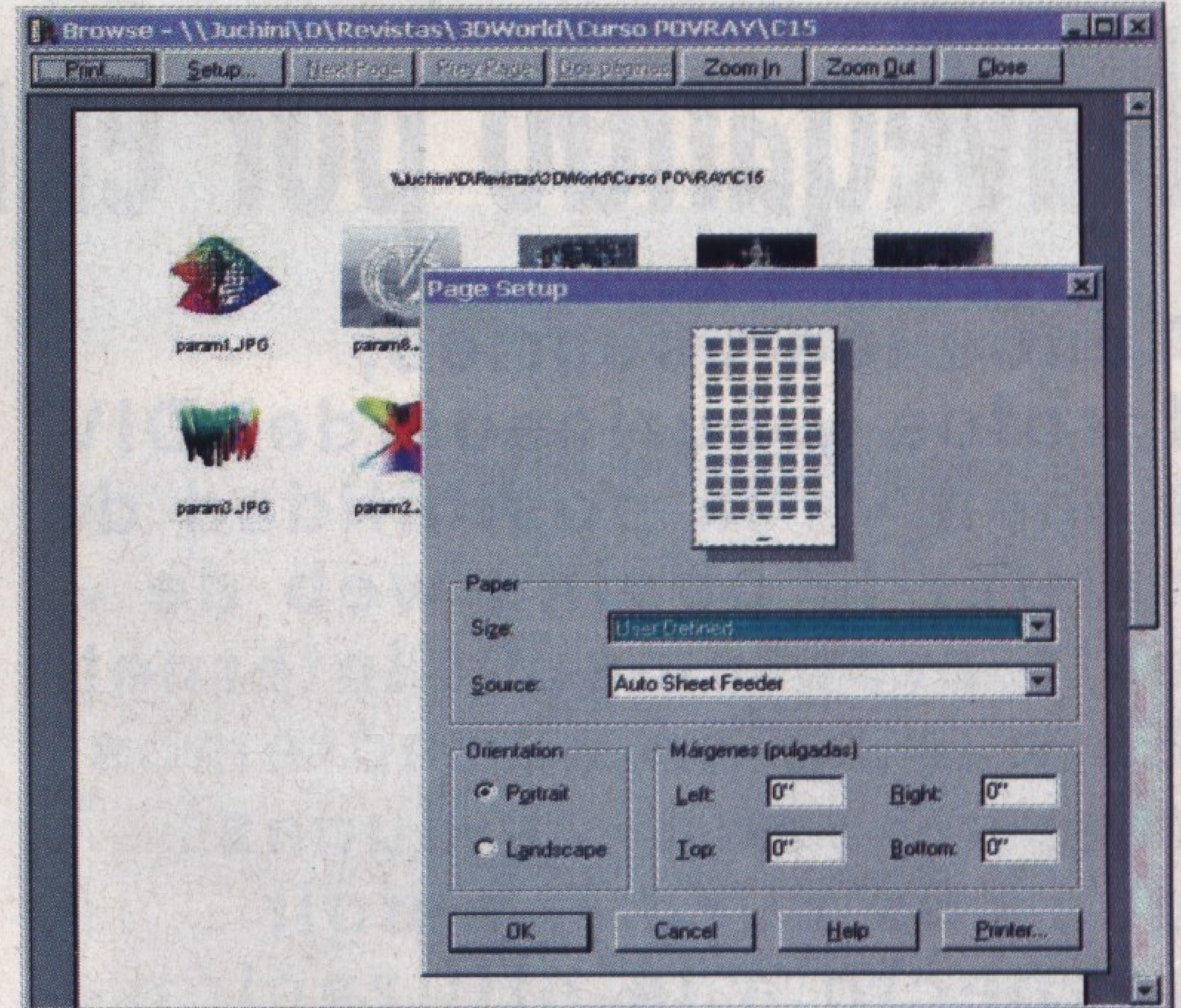


Figura 7. Composición con dos capturas correspondientes a Page Setup y Printer Preview cuando se utiliza la herramienta de previsualización Brower.

mientras que *Mirror* crea la vertical. *Rotate* rota toda la imagen un ángulo determinado y en un sentido de rotación concreto: horario o antihorario. Las siguientes opciones tienen que ver con las dimensiones de la imagen: *Add Border, Crop to selection, Canvas Size* y *Resize. Add border* crea un marco de color plano a la imagen. *Canvas Size* amplía el área de la imagen. *Resize* sirve para escalar la imagen.

Colors

Aquí aparecen efectos como *Colorize, Grey Scale, Negative Image, Posterize* o *Solarize* y funciones de histograma como *Equalize* o *Strech*.
En *Adjust* se maneja brillo, contraste, variaciones por canal, etc. También están en el menú *Colors* las opciones de paletas de color. Las últimas opciones que presenta permiten definir la resolución de *bits* por pixel mediante *Decrease Color Depth* o *Increase Color. Count Colors used* cuenta el número de colores de la paleta.

Layers

New Layer abre una ventana desde la cual se especificarán las características de la nueva capa. Tras pulsar en el botón *OK*, la capa se añadirá a la lista del programa. Para duplicar una capa se utiliza la opción *Duplicate*, mientras que para la eliminación se emplea *Delete*. Si se desea cambiar las propiedades de la capa, *Properties* abrirá una ventana idéntica a la que aparece cuando se crea la capa.
Arrange, View y *Merge* aparecen a continuación: *Arrange* cambia la posición de la capa en la lista de capas con las funciones: *Bring to Top* para ponerla la primera de la lista, *Move Up* para subirla una posición, *Move down* para bajarla una posición y *Send to bottom* para ponerla la primera contando desde abajo. *View* determina la visibilidad de la imagen en función de la capa seleccionada y *Merge* permite juntar todas las capas de dos formas: *Merge All* junta las capas sin distinción; *Merge Visible* sólo las visibles. En el desplegable de *Layers* aparece la lista de capas de la imagen seleccionada.

Selections

Desde aquí se manejan las áreas de selección. Cualquier selección se puede guardar de dos formas: como archivo de datos de selección en formato .SEL o como información de canal alfa, bajo los nombres: *Save to Disk, Save To Alpha Channel*.

Capture

Sirve para capturar imágenes. La primera opción, *Capture Setup*, abre una ventana desde donde configurar los parámetros. Para capturar, *Capture* y *Start* o SHIFT + C.

Browser

Cuando se pincha en *Browser*, dentro de *File*, aparece una ventana correspondiente al visualizador de imágenes. En la parte izquierda se muestra el árbol de navegación. Si se pincha en un directorio concreto que tenga imágenes (da igual el formato, ya que PSP es capaz de leer una infinidad de ellos) aparecerán a la derecha todas las imágenes en pequeño. Haciendo doble clic en una, se abrirá en el programa.

Mientras se encuentre abierta la ventana de *Browser*, la barra de menús cambia para ofrecer una serie de prestaciones a este previsualizador. Pinchando en *File* se despliega un nuevo menú con más opciones de las que aparecieron la primera vez. La primera de estas opciones es *Select*, con la que se puede buscar una imagen por 5 parámetros : anchura, altura, bits por pixel, nombre y extensión.
Sort configura el orden de búsqueda de *select*. Las siguientes opciones de importancia en *File* son *Page Setup, Printer Preview* y *Print. Page Setup* sirve para configurar el modo de presentación de la lista de imágenes del navegador, pero no para el entorno del programa. Desde aquí se configura el modo de presentación en papel: las imágenes son maquetadas por PSP 5 para ser impresas.

El proceso comenzará cuando se pinche en el botón *Start*. Desde *Preferences* se controla la mayoría de parámetros configurables en Paint Shop Pro 5.

Enrique Urbaneja

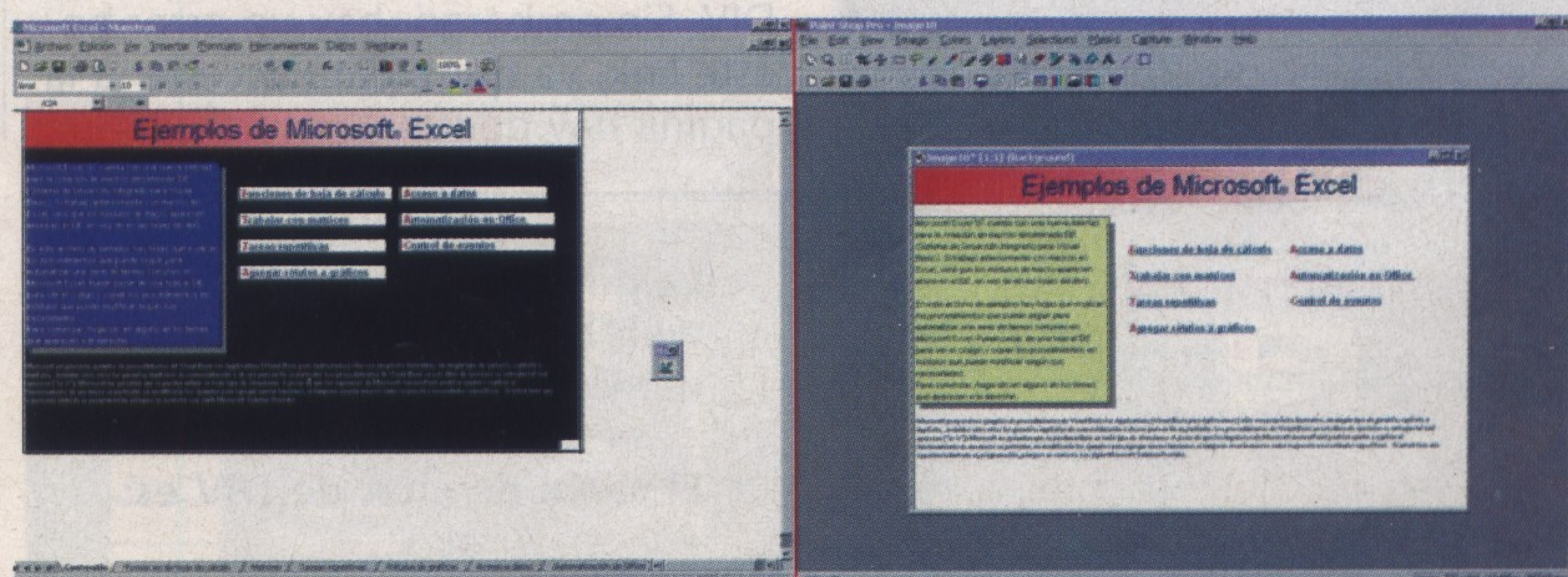


Figura 8. Page Setup y Printer Preview juntos.

Un fenómeno en la Red

Navegando por el mundo DIV

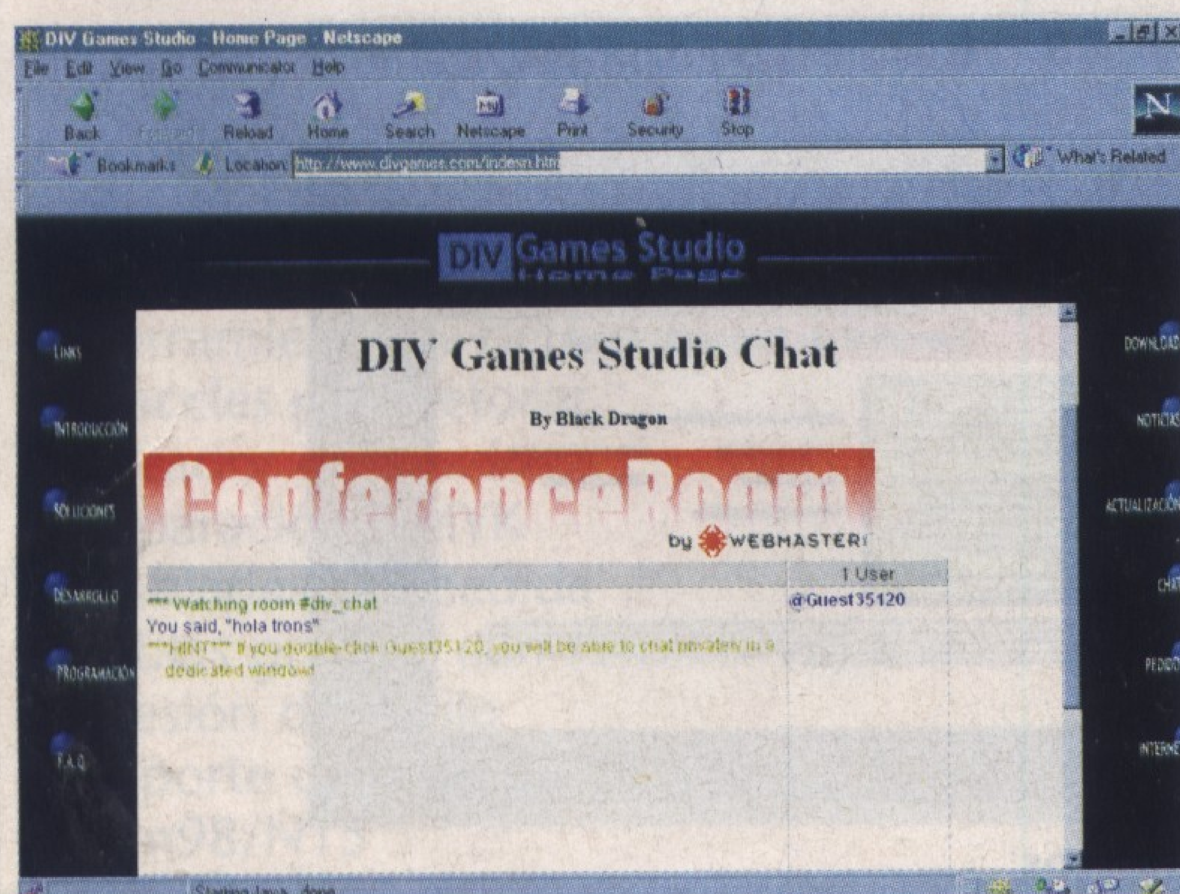
Todos los usuarios, forofos y curiosos del DIV tienen ya la posibilidad de visitar la página web de este programa en Internet. En ella no sólo tendremos respuestas a las dudas que nos puedan surgir respecto a la utilización de DIV, sino que podremos conocer las últimas novedades del programa y conectar con las páginas personales de los desarrolladores del mismo.



En la página web de DIV encontraremos una enorme cantidad de información sobre el mundo DIV, tanto a nivel usuario como a nivel neófito, ya que allí podremos conocer qué es el DIV.

Para ello, ofrecen un curso de programación, así como un servicio de respuesta a las dudas más frecuentes. El resultado es una página diseñada por y para los amantes de este programa, y del desarrollo de los videojuegos en general. Con un sencillo sistema de navegación por las página se puede ir accediendo a los diferentes contenidos de la misma, de manera intuitiva y rápida. Así, una serie de enlaces nos conducen a las diferentes secciones de la página oficial de DIV, con todas las respuestas, actualizaciones y demás novedades al respecto. También podremos encontrar una sección de download, con interesantes propuestas y utilidades que bajar.

Pero, sin duda, lo más interesante es la opción de acceder a las páginas personales de los desarrolladores de DIV. Para ello, iremos a la sección «Links» y se nos abrirá una ventana que contiene los hiperenlaces a las susodichas páginas.



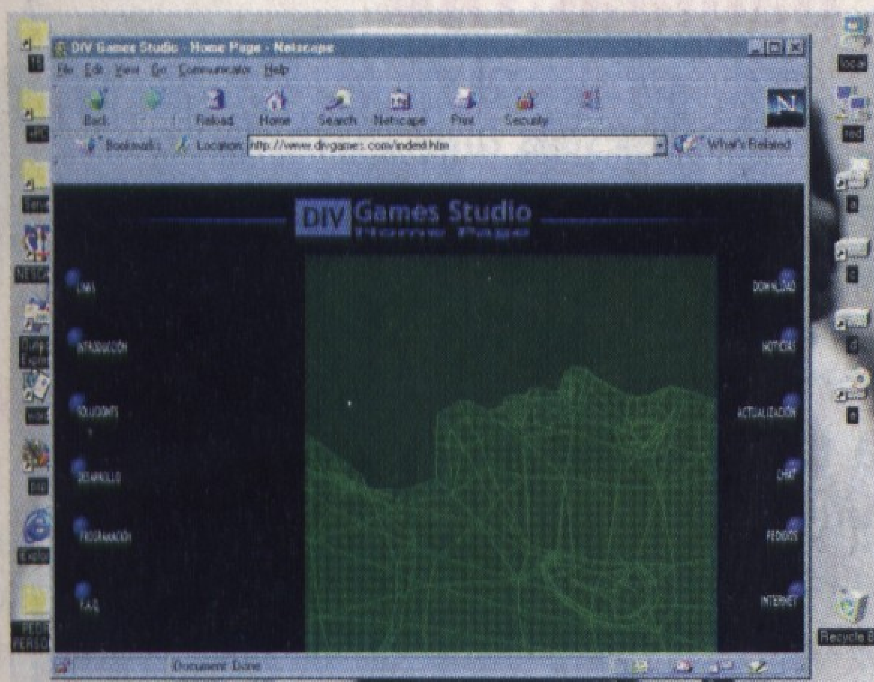
Páginas de los desarrolladores

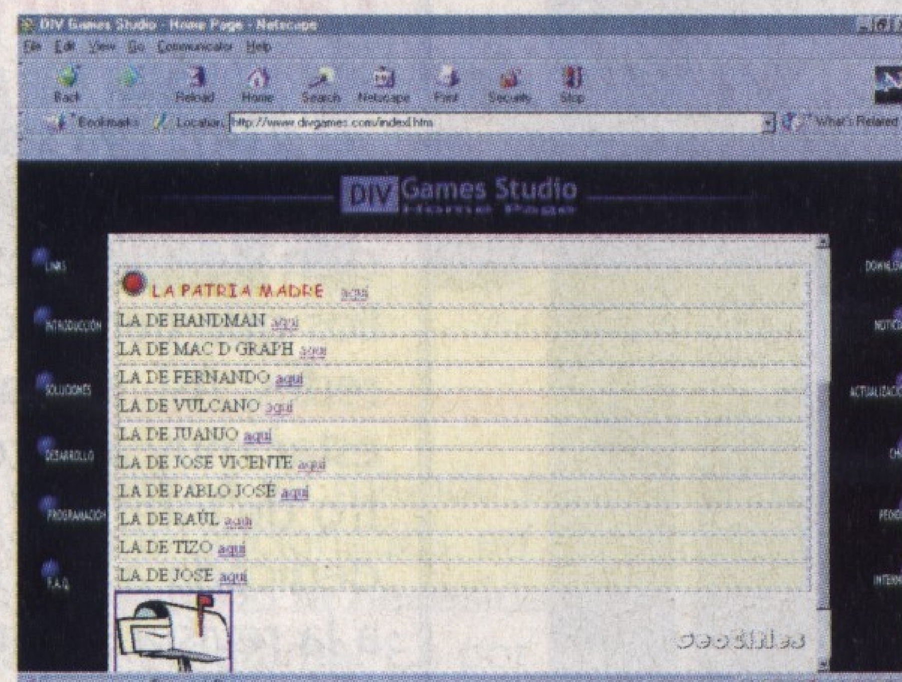
La primera página, cómo no, es la de «Tizo», Antonio Marchal, alma máter del grupo de desarrollo de DIV. Sin embargo, hay un error en este link, ya que , para acceder a la página hay que quitar «default.htm» del link. Estos son los problemas de las páginas en construcción, o los links conducen a otro sitio o a ninguno, como ocurre con algunos de los links, que nos llevan a páginas que ya no existen.

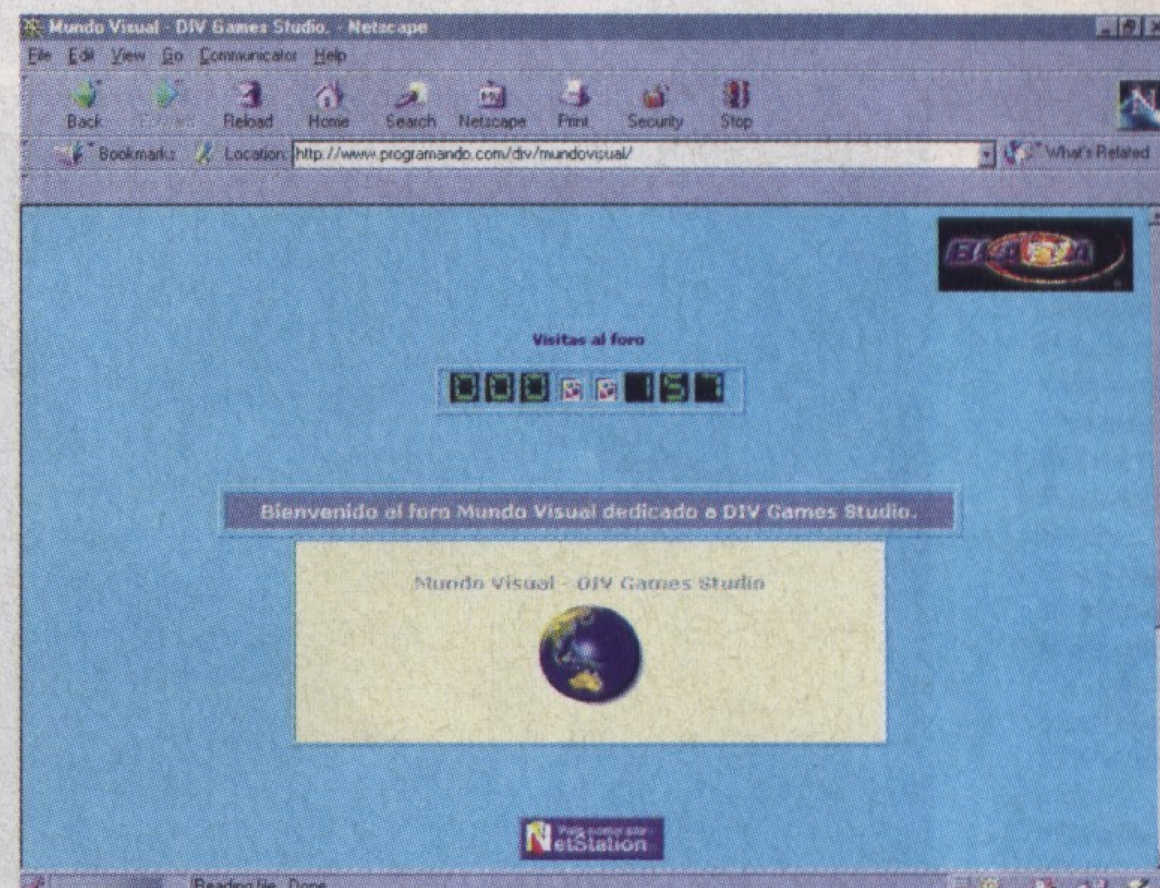
Para acceder a la web de «Tizo», copiamos la dirección sin «default.htm», y accederemos a la página de Tizosoft, donde encontraremos más enlaces, noticias y diversos downloads, como el de la máquina tragaperras. Desde ahí podremos acceder a más direcciones de Tizosoft, en Geocities y en Fortune City, donde podremos obtener más juegos en download y un avance de las próximas producciones de Tizosoft.

Pero no sólo encontraremos la página de Tizo en los links, también veremos las de otros desarrolladores, como la de José Vicente, una página muy trabajada y en constante crecimiento para ofrecer la máxima actualidad a los visitantes. Es una página muy dinámica, llena de movimiento de iconos y con un diseño que recuerda a la casa de J. F. Sebastian en Blade Runner, por la multitud de pequeñas cosas animadas que se ven en pantalla.Esta página también con-

- **La dirección de la página web de DIV es:** *www.divgames.com*
- **El canal de chat de DIV es:** *#div* en Arrakis *irc.arrakis.es*







tiene links a las páginas de sus compañeros de desarrollo de DIV.

La página de Iván García, de 15 años, también posee animaciones en los iconos y enlaces a otras páginas, además de presentar a su autor convenientemente.

En la página de Vulcano destacan, sobre todo, unas letras doradas giratorias que anuncian el contenido de la web. Enlaces, la presentación del programa y de los futuros proyectos y mucho humor conforman el contenido de esta página.

La página de Juanjo presenta noticias, enlaces, y consejos de utilización de DIV.

Desde Ceuta nos llega la web de Pablo José, que ofrece mucha información de las novedades del desarrollo de DIV.

DIV IRC. El chat del DIV

El canal chat de DIV se encuentra en ARRAKIS, al cual se accede en *irc.arrakis.es,* pudiendo luego entrar en el canal DIV. En la página de DIV encontraréis los requisitos para entrar en las conversaciones sin necesidad de tener que estar todo el día buscando a la gente.

0Así, se conciertan los debates públicos a las horas pares, 14:00, 16:00, etcétera, para no tener que esperar eternamente una entrada. Si a las dos no hay nadie, hasta las cuatro no hace falta ir a buscar, reservándose así las horas impares para las charlas privadas concertadas. En este foro se discute acerca de los trucos de programación, las novedades y las tendencias de la programación en DIV.

La página de José está muy trabajada, y ofrece muchos enlaces y noticias de lo último en DIV.

Otras cosas

En la página de DIV también encontraremos información sobre qué es el DIV, para lo que sólo tendremos que introducirnos en el enlace a la introducción, donde se explica el DIV para los que no saben exactamente lo que es.

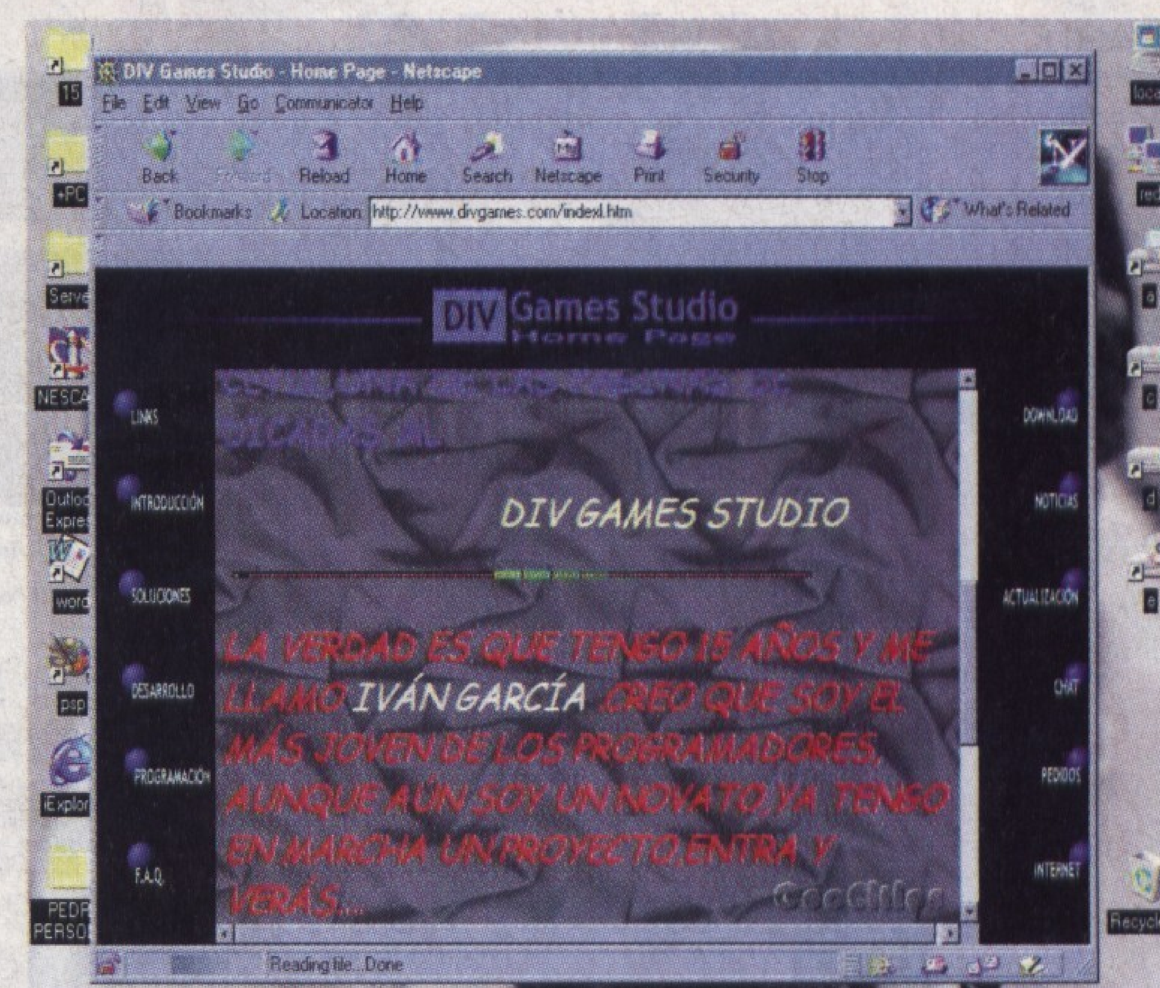
En la sección de soluciones se relata una fe de errores y se solucionan dudas, remitiéndonos al servicio técnico de Hammer Technologies, empresa desarrolladora del programa.

En *Desarrollo* sabremos exactamente dónde se encuentra el programa, los últimos errores detectados, las mejoras introducidas, etc. En definitiva, los avances en el diseño del programa.

El apartado *Programación* ofrece trucos y técnicas para programar en DIV, así como la posibilidad de que los usuarios envíen sus propios trucos y utilidades de programación en DIV.

La sección de F.A.Q.'s, (*Frecuently Asked Questions,* Preguntas Frecuentemente Planteadas) resuelve las principales dudas a la hora de usar este programa, Los problemas más frecuentes a la hora de instalar o los formatos no reconocidos aún por DIV.

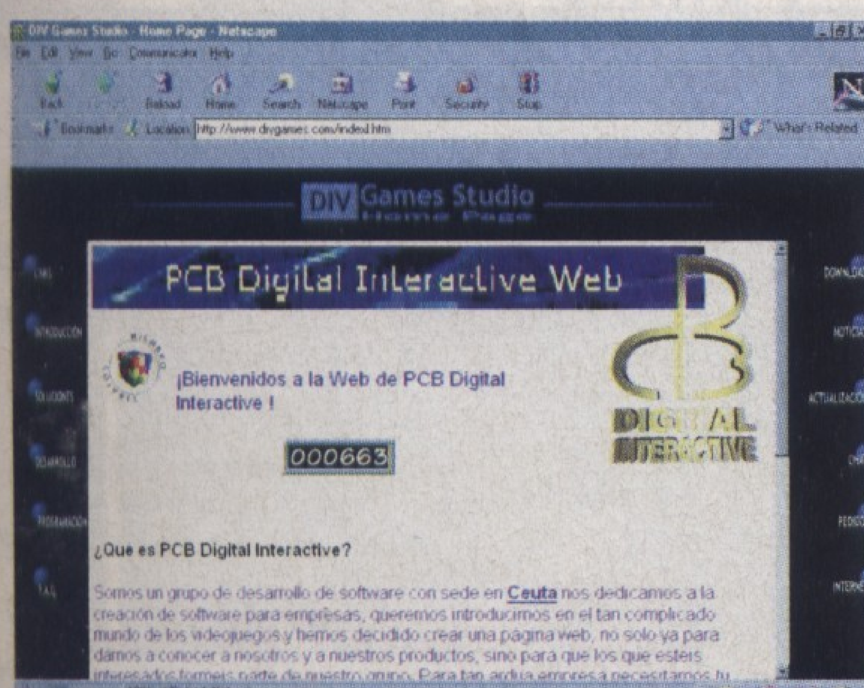
La zona de *Download* conduce a otras dos páginas, la oficial de descarga y la de aportaciones de usuarios. En la zona oficial hay una demo del Div, mientras que en la zona de aportaciones de los usuarios podemos encontrar juegos y otro tipo de utilidades que se pueden agradecer a la hora de programar en DIV.

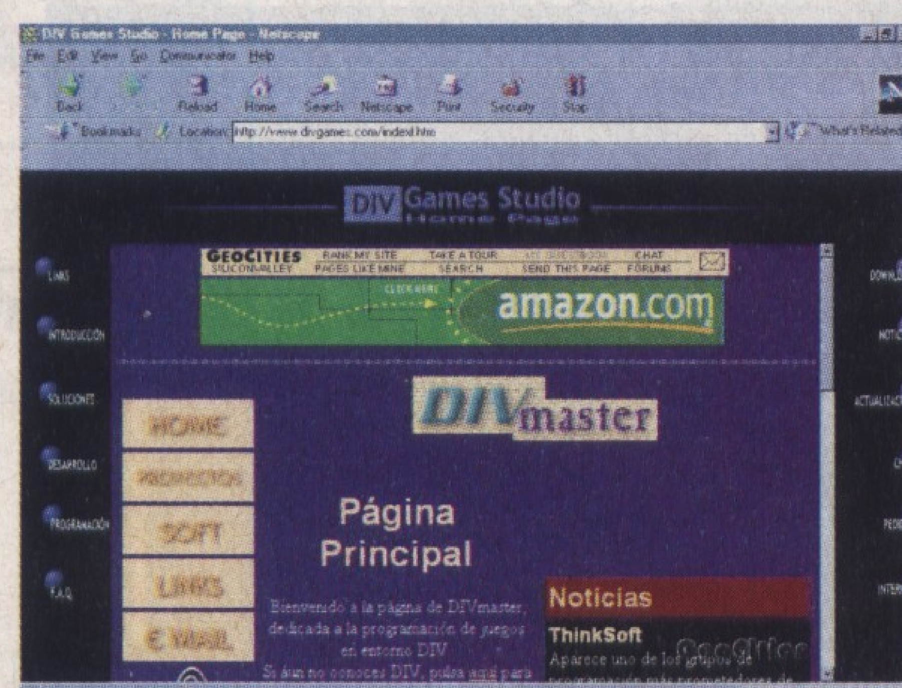


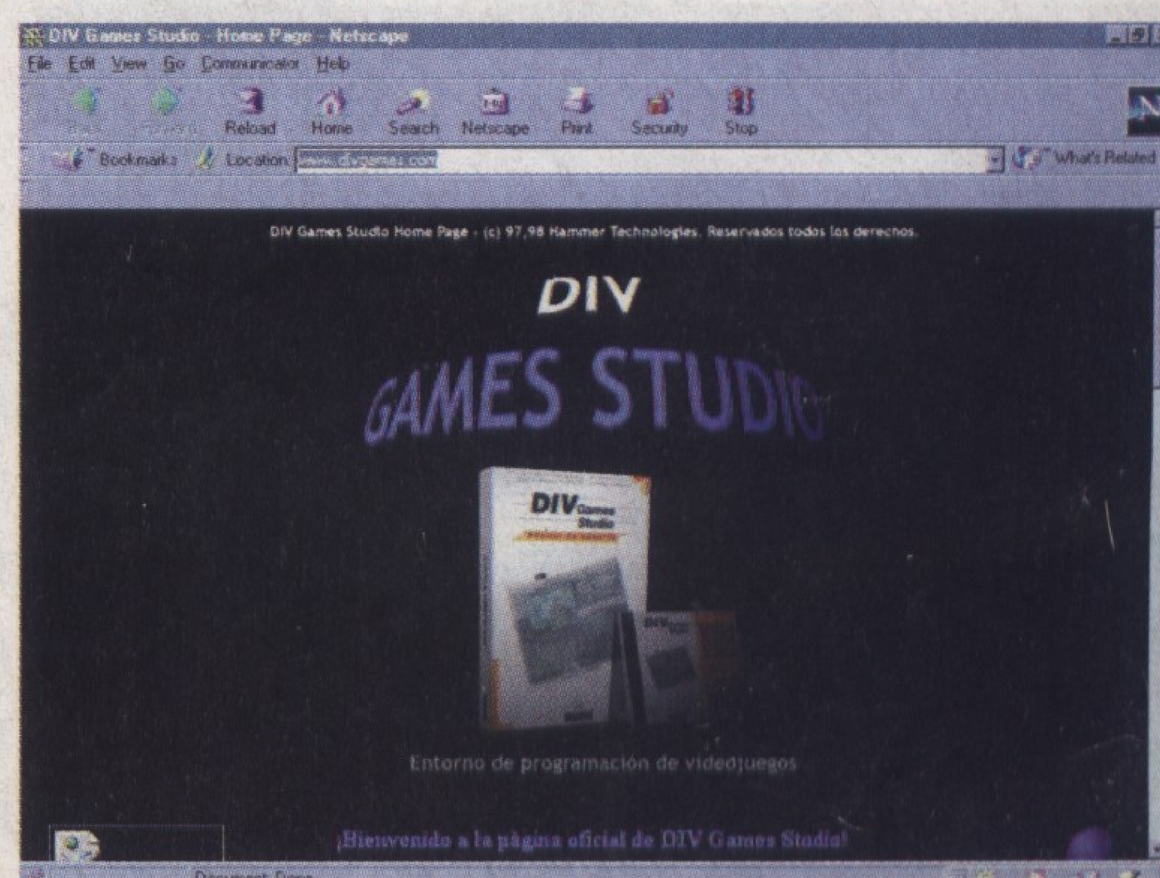
En la zona de noticias podemos encontrar el DIVIARIO, revista de información sobre el mundo DIV y las novedades de sus colaboradores, así como referencias a libros sobre DIV y otro tipo de informaciones de interés para los usuarios y entusiastas de este programa.

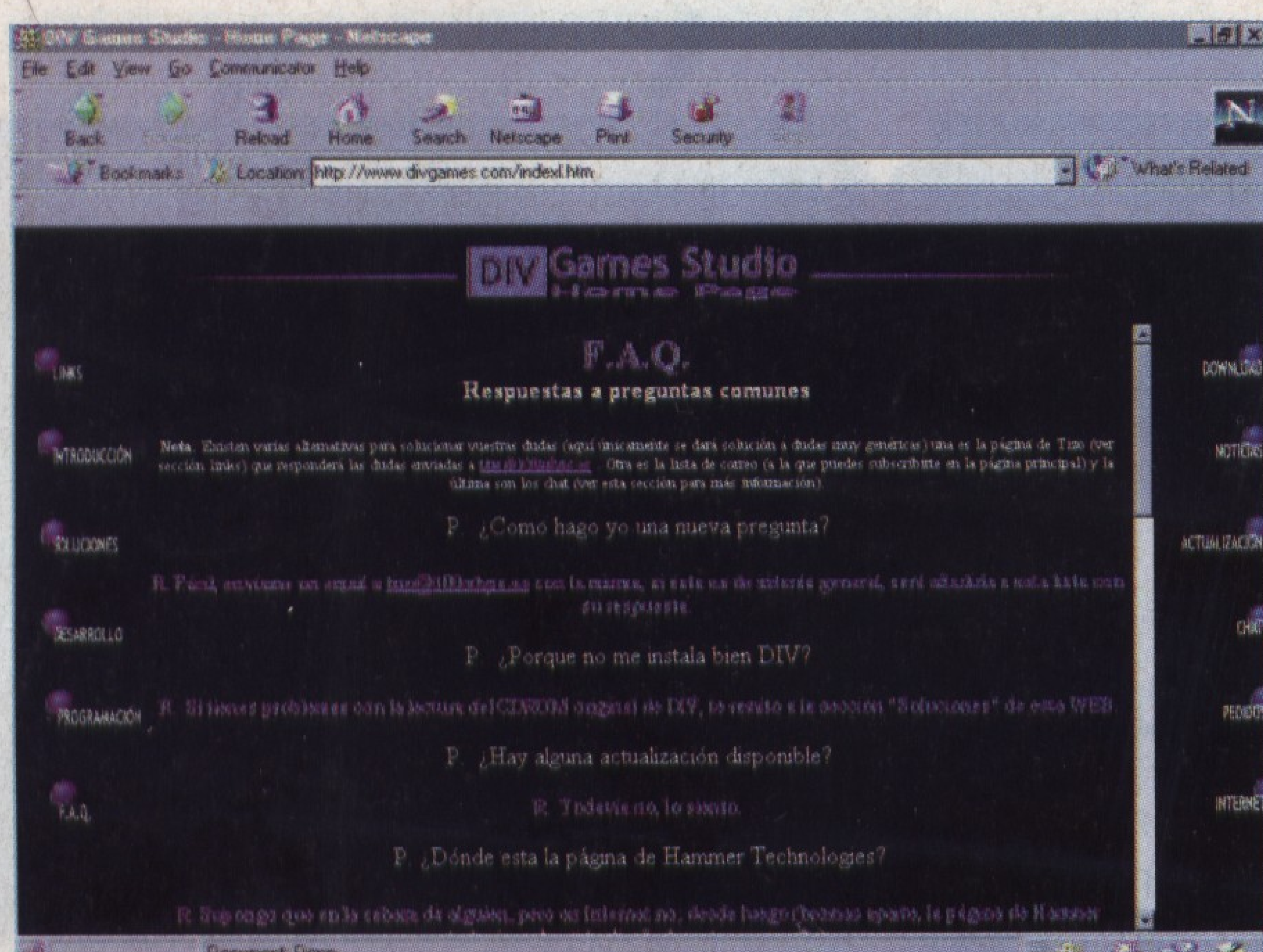
En la sección dedicada a la actualización de la web podemos ver un completo historial de todos los retoques efectuados tanto al diseño como al contenido de la página, ordenado en riguroso orden de actualidad.

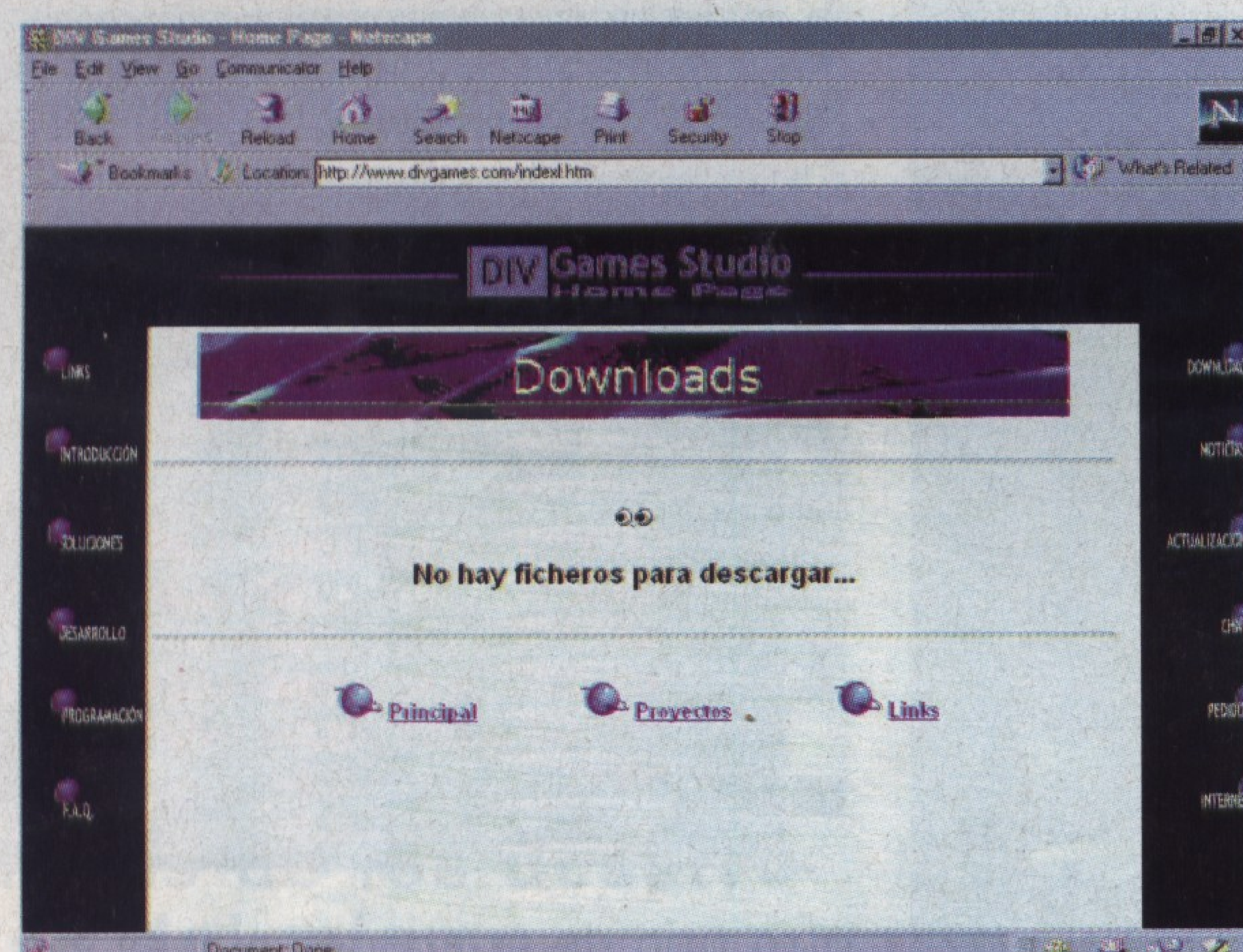
También podemos hablar directamente con el chat de DIV, mediante un enlace directo al canal DIV. Se recomienda leer antes las instrucciones de uso y funciona-











Las mejores páginas DIV

Como ya sabéis todos, a estas alturas de la revista, DIV manía desea contar con la máxima colaboración de los lectores. Y especialmente con la de aquellos que tienen una vida activa dentro de la Red de redes. Así, en cada edición de la revista vamos a estar muy atentos a todas vuestras evoluciones dentro de Internet. Aquellos que tengan páginas web dedicada a DIV no deben dudar en darlas a conocer a la revista, pera que desde aquí las demos a conocer a los demás lectores de la revista.

Nosotros mismos visitaremos vuestras páginas web y navegaremos a través de ellas para ver qué ofrecéis a los internautas. Cada vez aparecen más páginas dedicadas a DIV, y su vistosidad es cada vez mayor, de manera que no debéis dormiros en los laureles y seguir adelante con espíritu creativo.

La manera de ponerse en contacto con nosotros es muy sencilla. En primer lugar, puedes darlo a conocer en el canal de chat dedicado a DIV (se llama DIV y se encuentra en arrakis) o bien en la página web oficial de la herramienta, www.divgames.com. También puedes mandar un E-mail a nuestra revista (divmania@prensatecnica.com) en el que nos presentes tu página web. También puedes comunicarlo por correo tradicional, a la dirección de la revista, indicando en el sobre: "Páginas WEB". Nuestra dirección es:

C/ Alfonso Gómez, 42, nave 1-1-2
28037, Madrid, España

miento del canal, para no encontrarnos con el mismo vacío a la hora de charlar. Las horas pares son las de cita para las charlas públicas y las impares para las privadas.

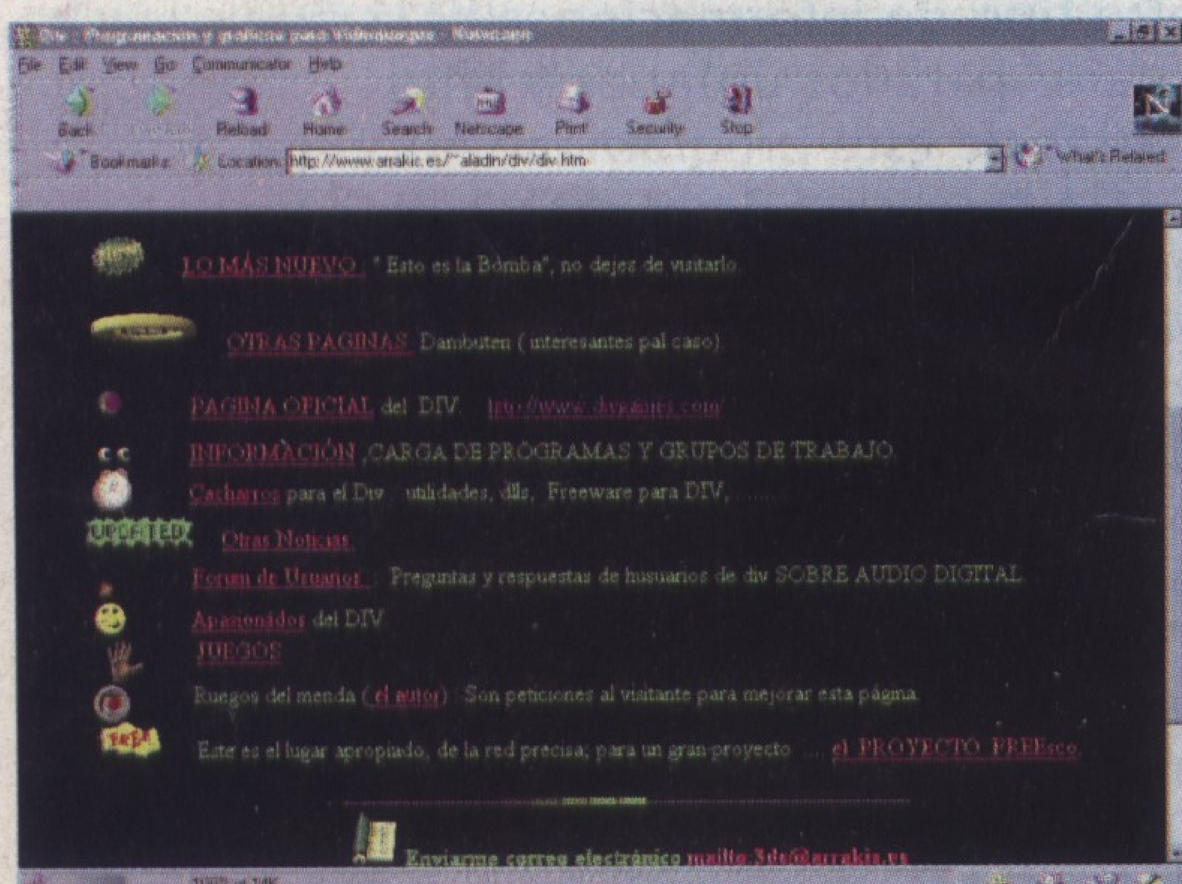
Desde la página web de DIV podemos pedir el programa, saber dónde adquirirlo y conocer el precio exacto del mismo (4.995 pesetas). Esto se puede ver «clicando» en la sección de pedidos de la página web.

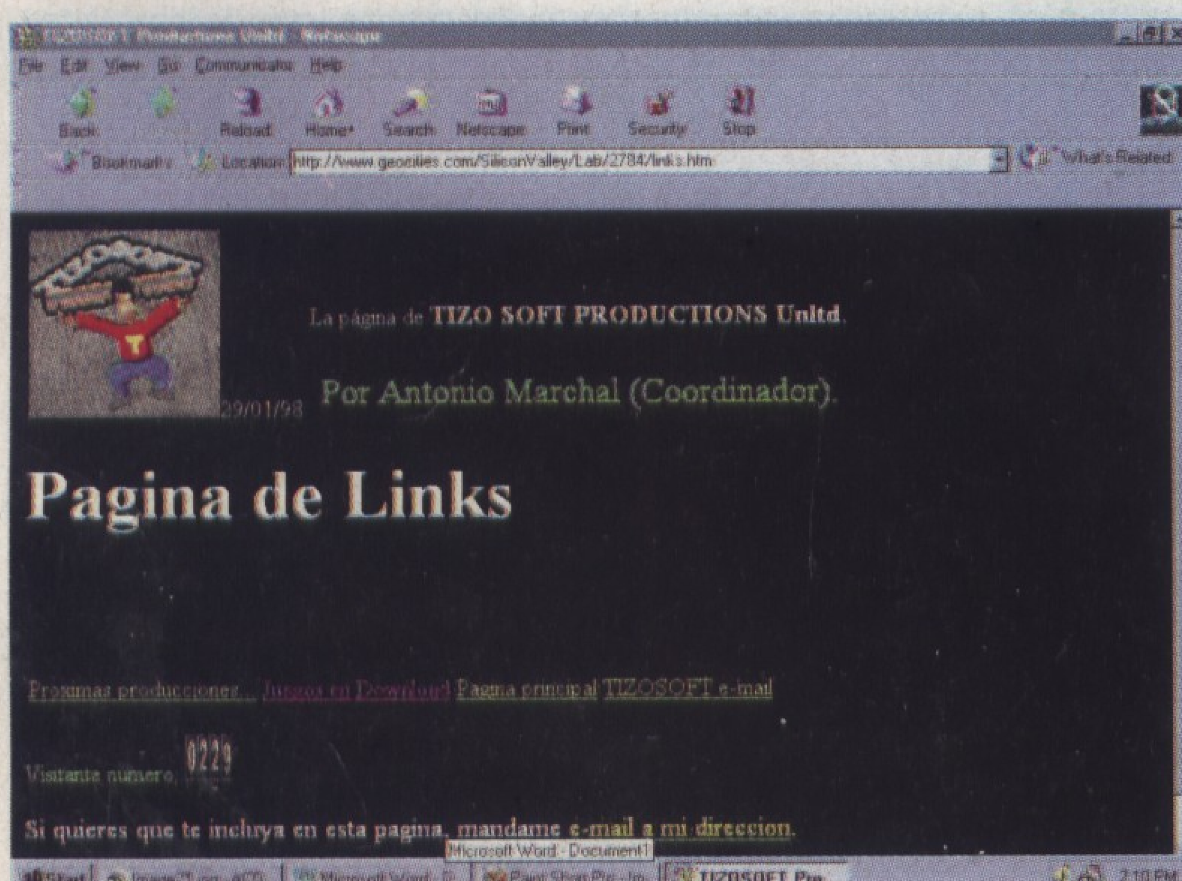
En la parte titulada Internet podemos acceder a enlaces con páginas de interés para los programadores de vídeojuegos, como buscadores, páginas de actualidad sobre los últimos proyectos que se encuentran en desarrollo en estos momentos, etcétera.

Conclusiones

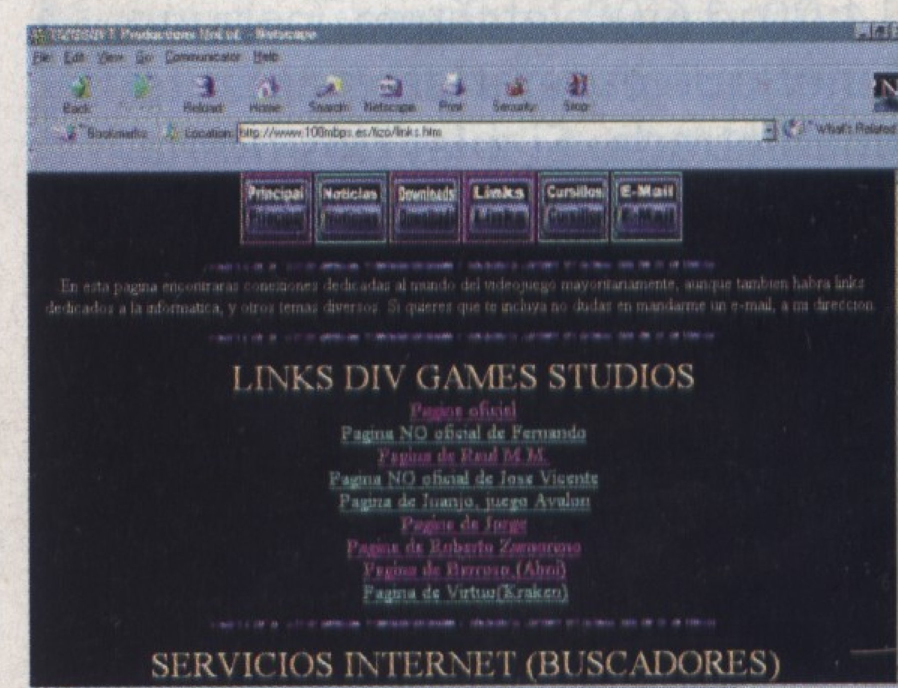
En definitiva, la web de DIV es un punto de encuentro obligado para los fanáticos de los videojuegos, de las últimas tendencias en programación y, cómo no, del DIV. La dirección de la misma es: *www.divgames.com.* Es la puerta del universo DIV para curiosos, profesionales y forofos de este programa.

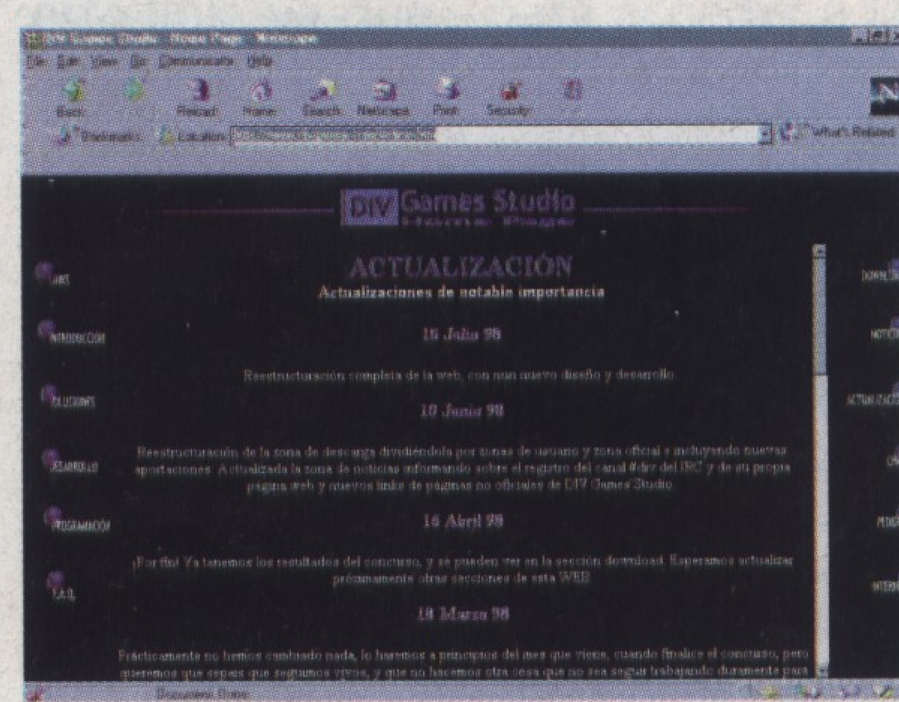
Pedro Solís

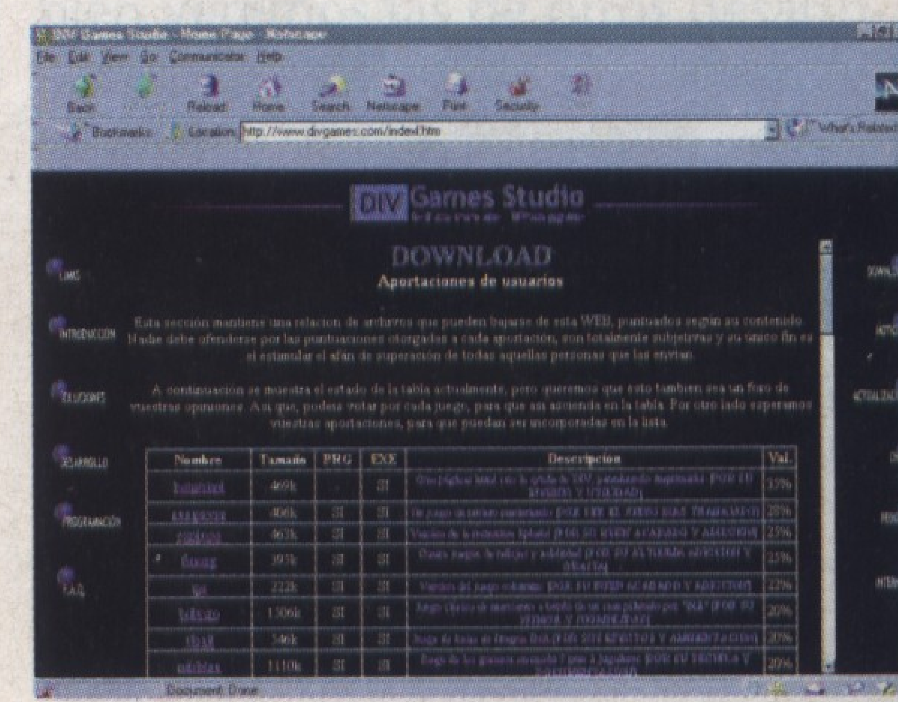












Primeros pasos con DIV

Aprende a manejarte en este entorno

DIV es la herramienta perfecta para desarrollar juegos, tanto para aquellos que no tienen conocimientos de programación general, como para profesionales que buscan un sistema sólido de desarrollo. En esta serie de artículos aprenderemos de forma práctica cómo crear ese juego que siempre soñó.

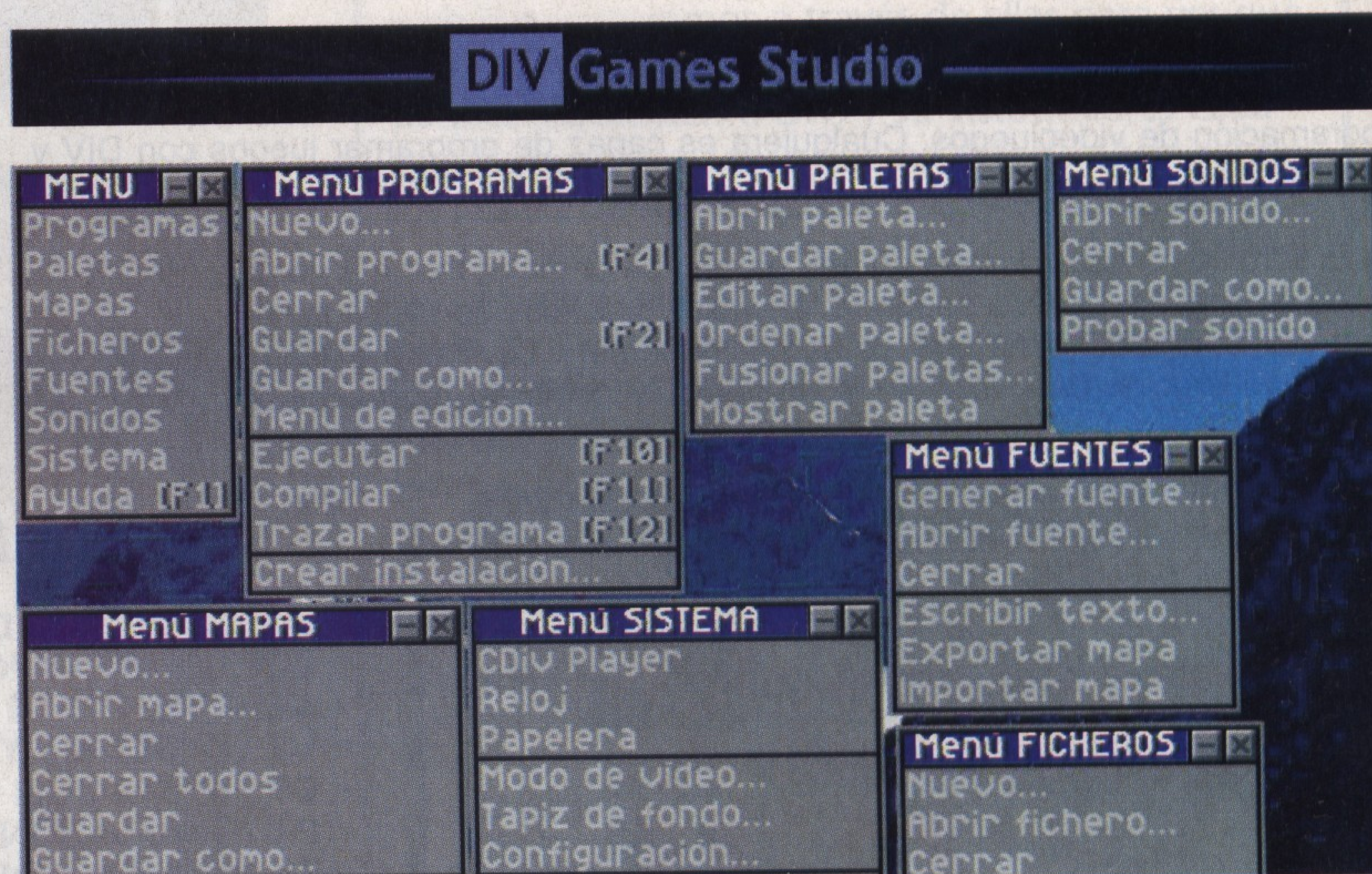


Imagen del entorno de DIV Games Studio con todas las opciones abiertas. Se puede observar su semejanza con el sistema de ventanas de Windows 95.



El desarrollo de cualquier programa o juego comienza con el conocimiento profundo del entorno en el que se está trabajando. Antes de comenzar con nuestro primer programa, analizaremos los puntos más importantes del entorno gráfico de DIV.

El entorno gráfico

DIV posee un sistema de menús en forma de ventanas parecido al de Windows 95, por lo que a los usuarios de este sistema operativo les será fácil adaptarse al entorno. Analizaremos una a una las opciones básicas del menú principal:

- Programas: es el menú destinado al manejo de ventanas de programa, es decir, aquellas donde irá el código que va a definir el comportamiento de nuestro juego, así como el control de la ejecución, depurado e instalación de ellos.

Las opciones básicas son *Nuevo*, *Abrir* (F4), *Cerrar*, *Guardar* (F2) y *Guardar como*: Gestiona el almacenamiento en disco y carga de los archivos de programa.

- *Menú de Edición*: Abre una ventana con *Copiar*, *Mover*, *Borrar*, *Buscar*, *Marcar Bloques*, *Reemplazar*, etc., con las que se modifican los programas.

Estructura principal de un programa de DIV

```
PROGRAM ejemplo

BEGIN
<sentencias>;
END
```

Datos

No sólo de sentencias vive el programador. Todo programa necesita de unos datos o valores numéricos que se almacenen en memoria, como por ejemplo los puntos conseguidos en un matamarcianos o el resultado en un partido de fútbol. A estos valores se les denomina datos. Hay diversos tipos de datos, pero en este primer capítulo sólo estudiaremos las variables, por ser el más sencillo.

Una variable es un valor numérico al que se le asigna un nombre, con el que nos referiremos a este valor. Para dar valores a las variables se recurre a una sentencia:

Variable = <número>;

Donde «número» es un entero cuyo valor puede oscilar entre _2.147.483.648 (no se pueden utilizar decimales). Una vez asignado el valor a la variable, se puede utilizar a dicha variable para referirse al valor almacenado en ella. Por ejemplo:

Puntos = 500;
Total = puntos + 200;

Por tanto la variable «Total» almacenará el valor 700 (500+200).

Podemos distinguir 3 tipos de datos según su área de repercusión, es decir, en qué parte del programa son conocidos:

Datos globales: Se puede acceder a ellos en cualquier punto del programa. Se declaran (asigna valores) antes de las sentencias principales, es decir, entre *PROGRAM* y *BEGIN*. El comienzo de la declaración comienza con *GLOBAL* (Cuadro 2).

Datos locales: Es un dato que en realidad son varios, pero denominados igual y con valor distinto en cada proceso.

Datos privados: Sólo puede accederse a ellos dentro del proceso en el que son declarados.

Para entender bien estos dos últimos tipos de datos, es necesario conocer cuál es la estructura de un proceso.

Ejemplos de sentencias usando operaciones con variables

```
PROGRAM ejemplo2
GLOBAL
Puntos = 0;
Total = 20;
BEGIN
Puntos = Puntos + 40;
Total = Puntos - Total;
END
```

Procesos

Como ya dijimos anteriormente, un proceso es todo objeto o gráfico que existe en un juego. Para definir el comportamiento de cada proceso, se crean unos bloques de sentencias. Cada uno de estos bloques representa a un tipo de proceso, es decir, que si queremos dibujar en la pantalla 10 coches iguales, basta con crear 10 procesos del mismo tipo y que dibujen un coche, con lo que tan sólo sería necesario un bloque de sentencias. Esto se verá más claro en nuestro programa de ejemplo. Para definir un bloque de sentencias se recurre a la estructura siguiente:

```
PROCESS <nombre del proceso> ( )
PRIVATE
<declaración de datos>;
...
BEGIN
<sentencias>;
...
END
```

Notamos la presencia de una sección *PRIVATE*, donde se declararán los datos privados, es decir, aquellos a los que sólo el proceso va a tener acceso y, por tanto, serán inaccesibles para el resto del programa.

Los procesos son objetos en pantalla con un código y variables asociados

La hora de la verdad: nuestro primer programa realizado con DIV

Una vez realizada esta introducción teórica, es hora de comenzar con la práctica. Vamos a realizar una sencilla animación, con lo que iremos aprendiendo las distintas funciones del lenguaje DIV según se vayan necesitando. Si no se tienen conocimientos previos de programación, tal vez le resulte un tanto escabroso. No hay problema si no entiende todo, con los ejemplos se irán asimilando los conceptos.

El programa que vamos a realizar es un simple desplazamiento de un coche por la pantalla. Para ello, en primer lugar es necesario tener las imágenes de coche. Podemos dibujarlo nosotros con el editor gráfico incorporado o cargar una imagen prediseñada de la librería de DIV Games Studio. Es preferible esta última opción, ya que aún no sabemos manejar el editor gráfico. Éstos son los pasos a seguir para obtener los gráficos:

Abrimos un archivo de mapas de coches, por ejemplo uno situado en el directorio *\Librería\coches*. Nosotros hemos elegido el archivo *coches1.map*. Para ello hemos seleccionado la opción *Abrir* del menú de mapas.

Nos aparecerá el mapa elegido abierto en una ventana. Hacemos doble clic sobre el fondo de la imagen para que ésta se abra con el editor gráfico.

Una vez en el editor gráfico, seleccionamos la opción de edición de bloques (simbolizado por unas tijeras y un recuadro).

Seleccionamos la imagen deseada haciendo clic dos veces, uno para comenzar la selección y otro para terminarla.

Acto seguido, pulsamos en la nueva opción que aparece en la barra de opciones simbolizada con una ventana. Salimos del editor pulsando el botón derecho del ratón.Con esto, habremos creado un nuevo mapa con las dimensiones de la selección.

Creamos un nuevo fichero, de nombre *ejemplo1.fpg* con la opción nuevo del menú *Ficheros*.

Arrastramos la ventana del nuevo mapa sobre la ventana del fichero (pulsando sobre la imagen y no sobre el título). Aparecerá una nueva ventana donde nos pedirá el código del mapa (elegiremos el 001) y una descripción del mapa (el que elijamos).

Una vez realizados estos pasos, podemos empezar a escribir nuestro código. Creemos un nuevo programa llamado *ejemplo1.prg*. En primer lugar vamos simplemente a colocar la imagen del coche en la pantalla. Para ello, vamos a crear un proceso llamado *coche()*. El código sería el siguiente:

```
PROGRAM ejemplo1;
BEGIN
set_mode(m640x480);
load_fpg("ejemplo1.fpg");
coche( );
LOOP
   FRAME;
END
END

PROCESS coche()
BEGIN
graph = 1;
x =100;
y=100;
LOOP
   FRAME;
END
END
```

Vamos a analizar las distintas sentencias que aparecen en el programa:

Set_Mode(<modo>): inicia el modo de vídeo que deseamos para el juego. Los valores posibles aparecen en el Cuadro 3. Si no se especifica esta sentencia al iniciar el programa, automáticamente se asigna el modo de vídeo 320x200. En nuestro juego utilizaremos el modo SVGA VESA de 640x480. Todos los modos tienen 256 colores.



Una vez realizada la selección de la imagen deseada, pulsamos la opción de cortar en ventana, situada en la esquina inferior derecha. Con esto creamos un nuevo mapa con la selección.

Modos de vídeo disponibles en DIV, todos tienen 256 colores

Con DIV disponemos de modos de vídeo que van desde resoluciones de 320x200 hasta 1024x768 con 256 colores.

VGA estándar	M320x200
Modo X	M320x240
	M320x400
	M360x240
	M360x360
	M376x282
SVGA VESA	M640x400
	M640x480
	M800x600
	M1024x768

- *Load_Fpg(<fichero>)*: Carga el fichero de imágenes que va a utilizar el programa. Se puede especificar una ruta para encontrar el archivo. Si no es así, lo buscará en el directorio por defecto (*.\FPG*).
- *FRAME*: Podríamos decir que es la orden de visualización. Indica cuándo un proceso está listo para ser visualizado. Cuando todos los procesos estén listos, la siguiente imagen se dibujará en pantalla. En el código principal también debe indicarse cuándo estamos listos para la siguiente visualización.
- *LOOP...END*: Es lo que denominamos un bucle, es decir, una repetición de sentencias. Todas las sentencias situadas entre estos 2 comandos, se repetirán infinitamente. Se suele utilizar junto con la sentencia *FRAME* para que la muestra de imágenes tenga continuidad.
- *Coche( )*: Crea un proceso del tipo coche, definido más abajo.

Dentro del proceso *Coche*, podemos encontrar tres variables de suma importancia. Son variables locales, es decir, propias de cada proceso. La variable *Graph* indica el número de la imagen del fichero cargado (ejemplo,*1.fpg*), que en este caso es 1, y que se corresponde con el gráfico anteriormente introducido en el fichero. Las variables *x,y* indican la posición de la pantalla en la que será dibujado todo proceso del tipo coche.

Ahora ya podemos ejecutar el programa pulsando F10. Pulse Alt+X para salir del juego.Efectivamente, tan solo sitúa la imagen en la esquina superior izquierda de la pantalla, pero ésta no se mueve. Para mover un proceso tan solo tenemos que variar sus variables *x,y* antes de mostrar la siguiente imagen o *FRAME*. Si añadimos entre *LOOP...END*, la sentencia...

x=x+1;

y ejecutamos, veremos como la imagen se desplaza hacia la derecha. Esto es porque cada vez que redibuja la pantalla, nuestro juego verifica la posición de cada proceso, incrementado en cada frame 1 píxel hacia la derecha.



Nuestro primer ejemplo mueve el coche a lo largo de la pantalla.

Ampliando nuestro ejemplo: Condiciones y códigos de teclado

Vamos a dotar a nuestro ejemplo de control mediante el teclado. Para ello, en primer lugar, necesitamos de las condiciones *IF*. Su estructura es la siguiente:

IF (<condición>)
<sentencias>
...
END

Su funcionamiento es sencillo. Si se cumple la condición entre paréntesis, se ejecutan las sentencias que la acompañan. Las posibles condiciones están indicadas en el cuadro 4.

- *Ejecutar* (F10): Compila y ejecuta el programa abierto.
- *Compilar* (F11): Compila el programa sin ejecutarlo.
- *Mapas*: opciones destinadas al manejo y creación de imágenes gráficas. Sus opciones son: Opciones de archivo (*Nuevo, Abrir...*): Crea, carga y guarda los archivos .MAP (extensión asociada a los archivos de mapas). Además, puede importarse archivos gráficos con formato PCX y BMP de 256 colores, automáticamente grabados como *.map*.
- *Reescalar*: Amplía o reduce el tamaño de la imagen. Se puede hacer de forma porcentual o expresando el tamaño en puntos. Podemos convertir una imagen en color a escala de grises.
- *Editar mapa*: Abre el editor gráfico, con el que podremos modificar o crear cualquier imagen que tengamos abierta.



- *Generador de explosiones*: Crea animaciones para explosiones.
- *Ficheros*: archivos de colecciones de imágenes con paleta común. Se arrastra cualquier ventana de mapa sobre la de fichero para añadir imágenes. Con un clic sobre una imagen se seleccionan o deseleccionan éstas. Posteriormente se pueden abrir todos en ventanas de mapas con *Cargar Marcados*, o seleccionando *Info* en la ventana de ficheros para obtener información asociada a cada mapa.
- *Sistema*: consta de varias opciones de control del entorno entre las que destacan herramientas como un reproductor de CDs, un reloj, una papelera, etc.
- *Ayuda*: hipertexto en el que podrá encontrar cualquier información acerca del entorno y el lenguaje de programación DIV Games Studio.

Además de estas opciones hay otras de importancia, como manejo de sonidos, fuentes y paletas o el trazador de programas, que analizaremos a lo largo del curso.

No hemos definido las opciones a fondo porque el artículo se eternizaría. Es aconsejable que exploremos por nuestra cuenta los entresijos de las opciones descritas. No obstante, algunas de ellas irán apareciendo en sucesivos artículos, a medida que se necesiten.

El lenguaje DIV

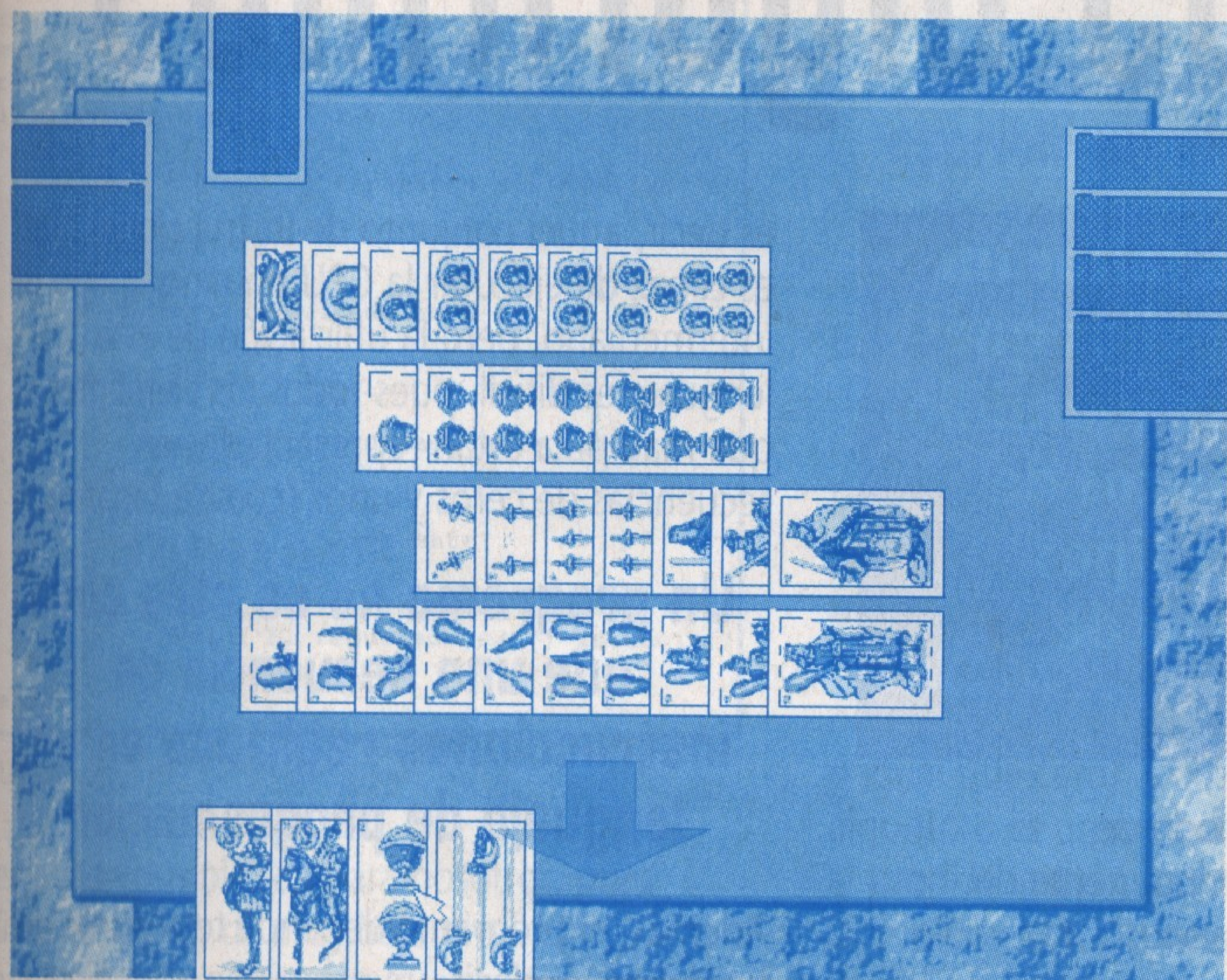
Otro aspecto necesario para desarrollar juegos con DIV es conocer su lenguaje de programación. Realizaremos una introducción e indicaremos algunos aspectos esenciales para poder comenzar con el desarrollo de nuestro primer ejemplo lo antes posible.

Pablo Trinidad

DIV developer

NÚMERO 1

Sumario



Área programadora

Dentro de una revista íntegramente dedicada a la programación de videojuegos no podía faltar una sección dedicada al desarrollo, una zona Developer. Así, estas dieciséis páginas están dedicadas a quienes quieran adentrarse en el fascinante mundo de la programación lúdica, sin quedarse sólo en DIV. En efecto, DIV es una excelente herramienta de trabajo, pero puede alcanzar una efectividad mucho mayor si se combina con otros lenguajes de programación.

Con este motivo, os ofrecemos, de entrada, tres cursos cuyos primeros artículos se encuentran en este número. El primero de ellos se centra en la programación básica, es decir, en los algoritmos con los que muchos de nosotros nos hemos educado. El segundo gira alrededor del mundo de la programación en C, y el tercero y último en el fundamental Ensamblador.

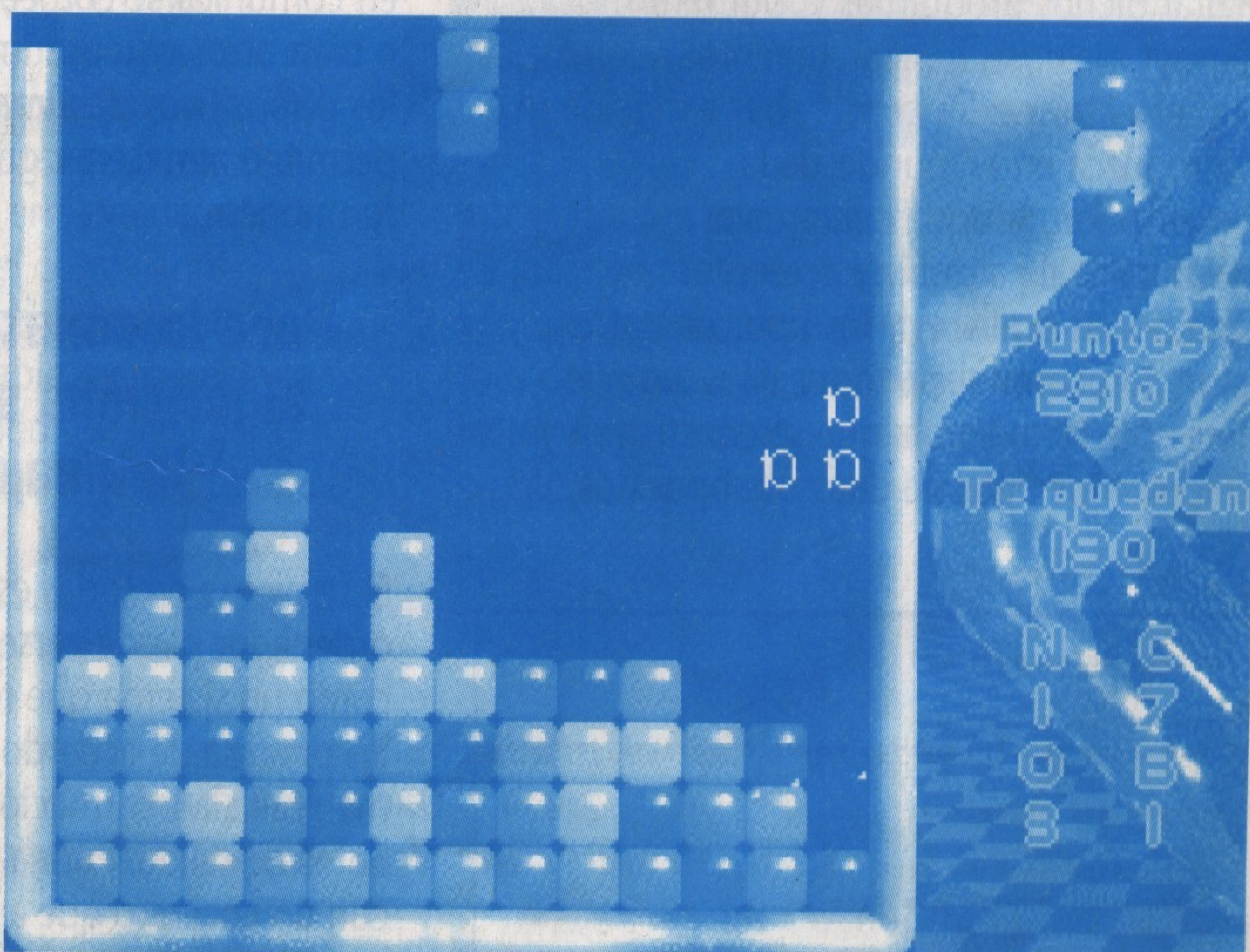
Los tres cursos tienen en común que parten de lo más básico, de manera que siguiéndolos cuidadosamente es posible profundizar en estos lenguajes de manera que sirvan de complemento ideal para DIV. Además, un programador que quiera avanzar no puede centrarse en un único lenguaje o corre el peligro de quedarse estancado.

Pero en estas páginas no sólo vamos a mostrar a los neófitos o iniciados los intrincados senderos de la programación. En efecto, como sabéis realizamos cada mes un concurso entre los lectores. Una de las claves del éxito de DIV es que permite a usuarios con escasos conocimientos de programación realizar viedeojuegos de mayor o menor calidad, dependiendo de las características de cada uno.

Así, en las casas de muchos de nuestros lectores, es decir, de muchos de vosotros, se esconden pequeñas joyas que nos gustaría dar a conocer. Con este motivo ofrecemos tres interesantes premios a los mejores programas que nos lleguen a la redacción. El primero de ellos consta de **25.000** pesetas, que recaerán en el programa que más éxito coseche entre nosotros. El segundo y el tercer premio tienen la misma cuantía, **20.000 pesetas**, lo cual tampoco está nada mal para quienes quieran dar a conocer sus pequeños juegos.

Para poder participar en el concurso no tienes más que escribir al E-mail de la revista: *divmania@prensatecnica.com* presentando tu juego, o bien enviarlo a la dirección:

C/ Alfonso Gómez, 42,
Nave 1-1-2
28037 Madrid, España



Destacamos

En nuestro CD de portada incluimos las siguientes demos, que han resultado ganadoras en el concurso que realizamos entre los lectores:

- **QA, un programa basado en clásicos de la categoría de *Columns*.**
- **Exploss, el juego del lector que ha conseguido el segundo puesto.**
- **Cinquillo, el clásico juego de cartas ahora dentro de nuestros PCs. El programa mantiene la fresca originalidad del juego.**

PROGRAMACION BASICA

Con ayuda de los algoritmos

Dentro de la sección DIV DEVELOPER no podría faltar un curso de programación general dedicado a aquellos que quieran empezar a dar sus primeros pasos en este mundillo. Se describe en este artículo los elementos básicos de la programación, clasificando los diferentes tipos de lenguajes, introduciendo los tipos básicos de datos y el concepto de variable.

Esta sección está pensada tanto para los lectores que se inician en el mundo de la programación como para aquellos que, conociendo ya algunos lenguajes, desean ampliar sus conocimientos en estructuras de datos o en algoritmos. También pretende ser una referencia para las personas que sigan en la revista los cursos de programación de un lenguaje particular, ya que estos cursos se centran más en las características propias del lenguaje que en algoritmos y estructuras de datos utilizables por cualquier lenguaje de programación.

Para la construcción de los diferentes algoritmos y estructuras de datos se utilizará un lenguaje de programación Pascal. Esto puede llevar a una pérdida de generalidad de los ejemplos incluidos en la sección; como contrapartida se obtendrá la posibilidad de ejecutar dichos programas y, por tanto, entender mejor su funcionamiento. Además, como el lector podrá comprobar a medida que se introduzca en el mundo de la programación, todos los lenguajes procedurales de alto nivel presentan instrucciones similares y los tipos de datos básicos son comunes a todos ellos. Lo que difiere es la forma en la que se construyen y manipulan los tipos de datos compuestos, aunque siempre hay un modo de implementarlos independientemente del lenguaje que se esté utilizando.

CUADRO 1.

```
Entrada del algoritmo:

  base de la potencia
  exponente (número entero mayor o igual a cero)

Proceso del algoritmo:

  Si el exponente es cero
    entonces la potencia vale 1
  Si el exponente es uno
    entonces la potencia vale la base
  Si el exponente es mayor que 1
    entonces la potencia es el número que
    resulta de multiplicar la base n-veces.

Salida del algoritmo:
  base elevado a exponente.
```

LENGUAJES

La estructura básica de un ordenador consiste en una unidad central de proceso (CPU), la memoria central, los dispositivos de entrada/salida y los dispositivos de almacenamiento secundario.

La CPU es capaz de ejecutar un pequeño conjunto de instrucciones elementales a una velocidad muy alta. La memoria central es la encargada de almacenar datos y programas, estos últimos indicarán al procesador el conjunto de instrucciones que tiene que aplicar a los datos. Los dispositivos de entrada/salida permiten la comunicación entre el ordenador y los diferentes elementos que están interesados en comunicarse con él, ya sean seres humanos o bien otros equipos informáticos. Por último, los dispositivos de almacenamiento secundario permiten almacenar gran cantidad de datos de forma permanente.

Para la construcción de los algoritmos se utilizará el lenguaje Pascal

Si se piensa en todas estas componentes, un ordenador puede verse, de forma simplificada, como un sistema que recibe datos de entrada y los procesa de acuerdo con el programa que manipula la información para producir nuevos datos de salida. Podría servir la típica analogía con un cocinero que a partir de los ingredientes apropiados (entrada) y una receta (programa) obtiene un plato elaborado (salida).

Por lo expuesto con anterioridad, para realizar la descripción del tratamiento que ha de darse a los datos, se dispone del lenguaje máquina, es decir, aquel conjunto de instrucciones que es capaz de ejecutar la CPU sin ningún tipo d intermediario, pero la utilización de este lenguaje es difícil, pues todos los elementos que manipula están representados en secuencias de ceros y unos (código binario).

Un programa no es otra cosa que una secuencia de instrucciones realizada en un lenguaje de programación

Además, el programa depende totalmente d hardware, es decir, un programa en código máquina no funcionará en otro ordenador q disponga de una CPU diferente, pues presentará un juego de instrucciones diferen La principal ventaja al utilizar este lenguaje es que se obtienen programas con una gran velocidad de ejecución y que aprovechan al máximo las características físicas del ordenad Para evitar las dificultades en la codificación programas en lenguaje máquina se crearon l lenguajes ensambladores. Estos, al igual que lenguaje máquina, dependen del hardware, pero a diferencia del anterior disponen de un sintaxis bien definida y significativa para el programador, ya que utiliza mnemotécnicos para representar las diferentes instrucciones. Como es evidente, el programa escrito en lenguaje Ensamblador no es directamente ejecutable por la CPU, por lo que necesita de un programa que traduzca las instrucciones lenguaje ensamblador a código máquina; a dicho programa se le denomina Ensamblado Las ventajas del lenguaje Ensamblador son la mismas que las del código máquina, pero se aumenta la legibilidad y la facilidad, en el grado que esto es posible, para realizar programas, aunque éstos seguirán dependiendo del hardware; es más, la utilización del lenguaje Ensamblador siempre lleva asociada un conocimiento profundo del hardware del sistema.

Gracias al lenguaje Ensamblador gran parte de los problemas que se plantean por la programación por medio del lenguaje máquin se han resuelto ya, pero los programas realizados en éste no son transportables, es decir, los programas no pueden ejecutarse en ordenadores con diferente procesador; ademá la programación en este lenguaje sigue siendo

```
program potencia;
var
 base, exponente, potencia: integer;
begin
 potencia:= 1;
 write('Introduzca la base >>>>');
 readln (base);
 write ('Introduzca el exponente >>>> ');
 readln(exponente);
 if exponente < 0
   then
     writeln ('Error el exponente es negativo')
   else
    if exponente = 0
      then
       writeln('El resultado es 1');
      else
       if exponente = 1
         then
           writeln('El resultado es ', base)
         else
          begin
           for y:= 1 to exponente
              do
               resultado:= resultado * base;
           writeln('El resultado es ', potencia);
          end;
 end.
```

CUADRO 2.

demasiado complicada para nosotros, pues utiliza conceptos de muy bajo nivel.

Para resolver los anteriores problemas se han creado los lenguajes de alto nivel, siendo su sintaxis mucho más cercana al lenguaje natural (aunque muy férrea), además de manejar una semántica más próxima a la forma de pensar del ser humano. Pero no todo son ventajas en la utilización de los lenguajes de alto nivel, pues el hardware deja de utilizarse de forma óptima, los programas aunque más fáciles de realizar dan como resultado programas ejecutables más grandes y más lentos que sus homólogos en lenguaje Ensamblador.

El algoritmo es un método general que se puede utilizar dentro de distintos lenguajes, en nuestro ejemplo Pascal

Al igual que ocurría con el lenguaje Ensamblador, los lenguajes de alto nivel necesitan de un programa que traduzca sus instrucciones a código máquina, estos programas se denominan traductores o compiladores. El primero de ellos interpreta instrucción a instrucción, ejecutándolas, por lo que no produce un programa ejecutable independiente del traductor, mientras que un compilador crea un nuevo archivo directamente ejecutable desde el sistema operativo de la máquina.

ALGORITMOS Y PROGRAMAS

Un programa no es más que una secuencia de instrucciones de un lenguaje de programación en particular, que permite resolver un problema. Resulta evidente que sin un método de resolución es imposible la realización de un programa; a dichos métodos se les denomina algoritmos. Un algoritmo no es más que una secuencia de pasos, sin ambigüedades, que seguidos uno a uno obtienen la solución del problema a resolver.

Es aconsejable diseñar y escribir un algoritmo antes de realizar un programa

Entre las características más destacables que debe tener un algoritmo se encuentra la de no contener ambigüedades, estar bien definido y la de ser finito (se llega a la solución en un número finito de pasos). En el cuadro 1 puede observarse un algoritmo expresado en lenguaje natural para calcular la potencia n-ésima de un número (supuesto que el exponente es mayor o igual que cero). Como se puede ver el algoritmo es un método general mientras que el programa del cuadro 2, que no es más que la implementación del algoritmo en lenguaje Pascal, es la expresión de dicho algoritmo para el caso particular de un lenguaje. El mismo algoritmo podría haber sido tomado como referencia para realizar la implementación en cualquier otro lenguaje de programación. Es aconsejable diseñar y escribir un algoritmo antes de realizar un programa ya que permite desarrollarlo con mayor sencillez y claridad. Los pasos a seguir para realizar un programa de pequeño tamaño y complejidad comenzarían por el diseño de un algoritmo para el problema que se pretende resolver; a continuación se procedería a editar un archivo de texto (programa fuente) con las instrucciones del programa; después, se pasaría a la fase de compilación para obtener a partir del programa fuente los errores sintácticos, si es que los hubiese. Tras corregir los errores de compilación que hayan podido surgir (el compilador muestra las líneas del programa fuente que son erróneas, tratando de diagnosticar el error), el compilador proporcionaría el programa objeto que contiene el código ejecutable, pero necesita posteriormente del enlazador (linker) para producir un fichero directamente ejecutable por el sistema operativo. Puesto que cada uno de los compiladores dispone de un entorno de programación que es diferente, no se hablará más del proceso de compilación, animando al lector a que consulte el manual correspondiente al compilador que haya seleccionado.

Un ejemplo

En el cuadro 3 se encuentra un programa realizado en Pascal que permite calcular el área de un círculo y la longitud de una circunferencia conociendo el radio. En Pascal todos los programas comienzan por la palabra reservada program; una palabra reservada del lenguaje no puede utilizarse para otro fin que no sea el indicado por el lenguaje (las palabras reservadas se encuentran en negrita)y a continuación se indica el nombre del programa; éste puede seleccionarlo el programador de igual forma que selecciona el nombre de los identificadores, además la regla sintáctica es la misma. A continuación, se han declarado tres variables de tipo real dentro de la sección dedicada para tal fin, que en lenguaje Pascal comienza por la palabra reservada var.

El tratamiento que se dará a la información viene recogido entre las palabras reservadas begin y end del programa. Entre las instrucciones que se encuentran en este bloque se pueden distinguir las fórmulas para el cálculo del área del círculo y la longitud de la circunferencia. Además, están acompañadas de operaciones de entrada/salida básicas (writeln y readln), que permiten obtener el valor para el radio desde el teclado y visualizar en pantalla el resultado obtenido por el programa.

En este ejemplo han aparecido nuevos elementos de un lenguaje de programación: las palabras reservadas. Se trata de palabras que el lenguaje reserva para su propia utilización; normalmente indican una construcción propia del lenguaje. Esto implica que el programador no podrá utilizar dichas palabras reservadas como identificadores.

También, puede observarse en el programa de ejemplo un comentario, la secuencia de caracteres delimitada por llaves; aunque esta secuencia está contenida en los programas fuentes el compilador las ignora a la hora de crear los correspondientes programas objeto; sin embargo cumplen la importante misión de autodocumentar el programa para así poder comprenderlo mejor. Es una buena costumbre incluir comentarios en los programas, pues ayudan a su mantenimiento y comprensión.

CUADRO 3.

```
program calculo_circulo;
var
 longitud, area, radio: real; {declaración de variables}
begin {instrucciones del programa}
 writeln('Introduzca el valor del radio ...  ');
 readln(radio);
 longitud:= 2*3.1416*radio; {3.1416 es un constante numérica}
 area:= 3.1416*radio*radio;
 writeln ('La longitud de la circunferencia es ....' , longitud);
 writeln('El área mide... ', area);
end.
```

Curso de iniciación en C

En estas páginas empieza lo que es el primer capítulo del curso de lenguaje C, el lenguaje de programación usado por la mayoría de programadores que se caracteriza, principalmente, por tener una sintaxis robusta, ser muy flexible y poseer tanto la facilidad de programación de un lenguaje de alto nivel como la potencia del ensamblador.

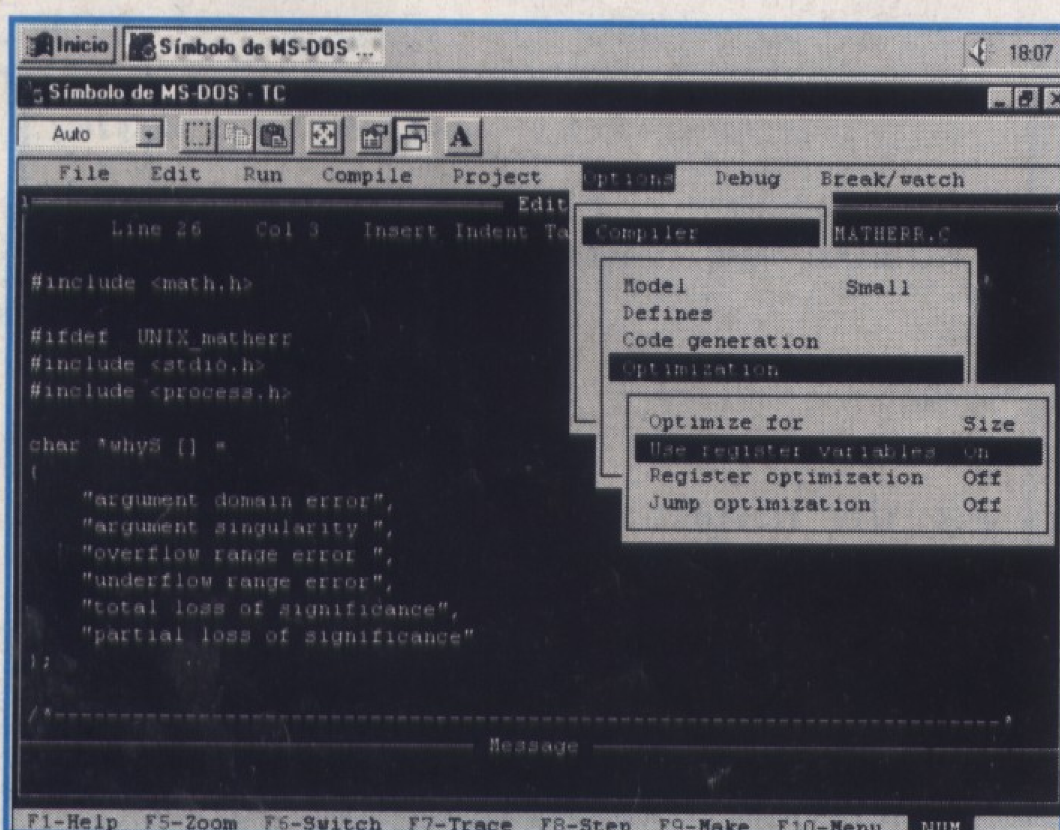
El porqué hacer un curso de C se explica rápidamente. Porque este lenguaje se ha convertido, en los últimos años, en el lenguaje preferido por todos los principales programadores del mundo entero y ha sido adoptado por el sistema operativo Windows como estándar de programación de esta plataforma.

El lenguaje C es el único que proporciona las ventajas de los lenguajes de alto nivel, como pueden ser BASIC o Pascal (otros dos lenguajes muy conocidos y populares desde hace mucho tiempo), a la vez que permite también poder ejercer un profundo control del hardware y de los periféricos instalados en el sistema, característica que comparte sólo con el propio ensamblador.

Además de lo dicho, el C tiene otras ventajas, ya que los mejores compiladores de C generan un código muy optimizado y rápido (los programas ocupan poco espacio físico y se ejecutan rápidamete).

Por otro lado, C es un lenguaje muy estructurado (característica que comparte con el Pascal), y su sintaxis hace fácil escribir programas grandes y modulables sin que estos pierdan la estructura general, lo que a la larga ahorra horas de trabajo al programador al realizar retoques, ampliaciones y/o modificaciones de secciones de programa.

Figura 1. El clasico Turbo C.



Por último, decir que al ser un lenguaje estándar de todas las principales plataformas (Pc, Mac, etc...), tenemos que un programa realizado en un PC lo podremos trasladar a otro ordenador del tipo que sea sin ningún problema.

ORIGENES DEL C

El lenguaje C fue creado por Dennis Ritchie, en Bell Laboratories, a principios de los años setenta y fue resultado de la evolución del por entonces lenguaje B.

En el año 1978, Brian Kerninghan y Dennis Ritchie publicaron el libro *The C programming Languaje* que lo hizo conocer al mundo y que puso sus bases como lenguaje.

Un lenguaje de programación que ya ha cumplido los veinte años mantiene su vigencia

Actualmente, existe un estándar de este lenguaje, creado por la *American Natrional Estandar Institute*, el conocido como ANSI C, y con el cual son compatibles la mayoría de compiladores (incluyendo los citados anteriormente), por lo que si escribimos un programa C para PC que cumpla todas las normas de este lenguaje establecidas en el ANSI C, este mismo programa podrá ser compilado y ejecutado sin cambios en cualquier otro ordenador mientras este posea un compilador compatible ANSI C.

QUE SE NECESITA EN EL CURSO

Primero de todo, se debe disponer de un PCcompatible, ya sea un XT, AT, o PS/2 ó cualquier modelo superior con todo el hardware mínimo instalado (monitor, unidades de disco, etc...).

El sistema debe tener instalado el sistema operativo DOS (MS-DOS, PC-DOS...), o uno que sea compatible con él, como es Windows 95, donde se pueden ejecutar programas D y como herramienta necesitaremos lo que s llama un compilador de C, que es el progra que nos permitirá poder ejecutar los programas C que escribamos.

Compiladores de C para MS-DOS hay much aunque los más usados y conocidos son los clásicos Quick C de Microsoft, el Microsoft el Turbo C de la casa Borland. En general to los compiladores son buenos, e incluso hay actualmente alguno que es completamente gratuito, el famoso DJGPP, que se puede conseguir por Internet desde múltiples luga o de algún CD-ROM de alguna que otra publicación informática.

¿QUE HACE EL COMPILADOR?

Una vez que el usuario ha escrito un progra en lenguaje C usando un editor de texto cualquiera, tenemos que éste no es más qu un fichero ASCII, un fichero de texto con el que el ordenador no puede hacer nada.

La función del compilador es la de encarga de coger nuestro programa (fuente en lenguaje informática) y convertirlo (lo cual s llama *compilar* en informática) a lenguaje binario (código máquina o ensamblador se como quiera llamarse), que sí es ejecutable directamente en nuestro ordenador.

Además de compilar, los compiladores tamb permiten poder encontrar errores sintáctico nuestros fuentes rápidamente y errores grav com el uso de variables inexistentes, uso de recursos no disponibles, etc...

ESTRUCTURA BASICA DE UN PROGRAMA C

Para aquellos que no estén familiarizados co el lenguaje C o incluso con ningún otro, cuando vean un programa escrito en C por primera vez, seguramente lo encontrarán complicado, como si fuese un listado lleno d extraños símbolos y largas líneas de código. principio, quizá les hará creer que es muy difícil poder llegar a comprenderlo y aprenderlo. Pero no hay por qué preocupar en realidad, no es más enrevesado que cualquier otro lenguaje y todo se limita a practicar y tener paciencia.

UN EJEMPLO DE PROGRAMA C

Primero de todo, decir que un programa en puede ser muy simple, teniendo que con ta

sólo cuatro líneas de programa se puede realizar un ejemplo completo y sencillo como es el siguiente:

```
main( )
{
 printf("hola");
}
```

EXPLICANDO LAS LINEAS

Primero de todo nos encontramos con la línea de programa *main*( ).
Todos los programas en C contienen esta línea (lo que se llama una función), que indica el inicio del propio programa. A partir de esta línea de programa, todo el código del programa que queramos incluir deberemos insertarlo entre los dos corchetes que hay a continuación del propio *main( )*.

Es fundamental la función del compilador. Se encarga de conseguir que nuestro fichero de texto ASCII se convierta en un lenguaje binario legible

En el ejemplo que se acaba de dar esto se ve claramente, ya que la única instrucción de programa (función) que se ha incluido como ejemplo se encuentra, como se puede comprobar, insertada entre los dos corchetes (de apertura y cerrado).

¿QUE ES UNA FUNCION?

Se ha dicho que *main( )* es una función (*main* = principal, función principal).
En lenguaje C todo en un programa está dividido en unidades llamadas funciones.
Una función es como un pequeño programa y todas las funciones tienen la misma sintaxis escrita, que consta de un nombre seguido por un paréntesis (que puede estar o no vacío).

LOS DELIMITADORES

Como se ha dicho, los corchetes indican el principio y final de la función tras la cual se escriben (del programa en el caso de *main*). Normalmente, los programadores llaman a los corchetes delimitadores.

LA INSTRUCCION DEL PROGRAMA

En el ejemplo que se ha dado, se ha incluido una instrucción entre dos delimitadores.
Si el lector se fija, la instrucción acaba con un punto y coma, lo cual no es casual ni un caso puntual, sino que es una norma del lenguaje C que todas las funciones (instrucciones) deben acabar por punto y coma (si no se ponen, el compilador nos dará un error cuando intentemos *compilar* nuestro programa para ejecutarlo).

Estilo de programación

Un programa en C se puede escribir de muchas formas, ya que se pueden incluir en una línea tantas instrucciones y espacios como se quiera teniendo, de esta forma, que es posible escribir el ejemplo dado anteriormente como sigue:

main() { printf("hola"); }

Para el compilador, esta línea de código, que es equivalente a las cuatro que habíamos escrito antes, es, en contenido, idéntica a del anterior ejemplo, lo que abre la posibilidad al programador a que pueda escribir los programas con el aspecto estético que quiera, aunque, como es lógico, hay unas normas generales que utilizan casi todos los programadores que hacen más legibles y entendibles a los programas.

También es importante que el lector sepa que en C se diferencian entre mayúsculas y minúsculas (lo cual no pasa por ejemplo en BASIC), y por lo tanto, si se escribe la misma instrucción *printf* como *Printf* o *PRINTF* por ejemplo, el compilador considerará que estamos indicando otra función y dará error, por lo que deberemos tenerlo en cuenta siempre a la hora de programar en C.
Además de todo lo dicho, señalar que esta instrucción en concreto, (*printf*), sirve principalmente para escribir un texto en pantalla, e incluye un parámetro entre los paréntesis y contenido entre comillas que es el propio texto que queremos que escriba.

LA CADENA "HOLA"

El texto incluido como parámetro de la instrucción *printf* es un ejemplo de lo que en programación (e informática en general), se llama una cadena.
En lenguaje C las cadenas siempre van escritas entre comillas, lo cual es la manera que tiene definido el compilador para poder reconocerlas (encontrarlas).

ERRORES DE COMPILACION

Si al copiar el programa de ejemplo se ha cometido algún error, como sería, por ejemplo, olvidarse de unas comillas de la cadena "hola", omitir un corchete, escribir *printf* con alguna mayúscula o cualquier otro error, al intentar compilar el programa, el compilador no concluirá la operación y nos informará del error.

Una de las normas básicas de C es que todas las instrucciones deben acabar por un ";". Si no, el compilador dará un error

Los errores que nos aparecen con este lenguaje cambian en la forma de ser descritos según el compilador que se esté utilizando aunque, normalmente, el propio mensaje de error nos dará una información suficiente como para que podamos encontrar el error que se ha producido dentro del código fuente (es decir, en el texto de programa).

Dicho todo esto, tenemos, por ejemplo, que si hubiésemos escrito el programa como sigue:

```
main()
{
printf("hola);
}
```

Al intentar compilarlo, podríamos obtener un error del estilo del que señalamos a continuación: *"Systax error: missing ")" before "}"*, que traducido sería: *Error de sintaxis, se ha omitido ")" antes de "}"*, con lo que se puede deducir que el error se encuentra en algún punto de los paréntesis (en este caso, las comillas de la cadena "hola").
No nos podemos ir sin decir a este respecto que hasta que el compilador no realiza el compilado entero sin ningún tipo de error. Éste no creará la versión ejecutable de nuestro programa y, por lo tanto, no podremos ejecutarlo y usarlo.

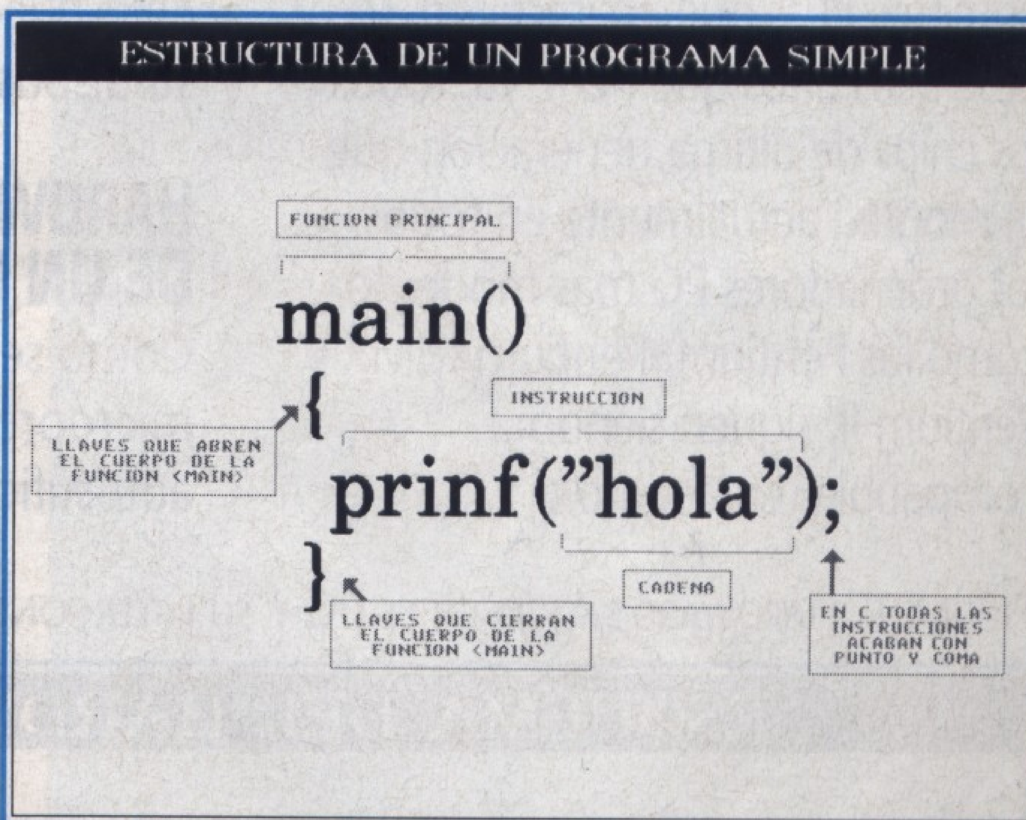


FIGURA 2. ESQUEMA QUE MUESTRA LAS PARTES DEL PROGRAMA QUE HEMOS TOMADO COMO EJEMPLO.

Conclusión

Con lo explicado en este capítulo, se pueden dar por explicados los conceptos más elementales relacionados con el lenguaje C. En el próximo capítulo se profundizará, entre otras cosas, en el tema de las variables, fundamentales de entender y saber usar.
Antes de acabar, sólo recordar que para preguntas al autor, el lector puede mandar un E-Mail a la dirección <erde@arrakis.es> o hacerlo en la web: www.arrakis.es/~erde.

Primeros pasos en el ensamblador

El ensamblador es el lenguaje con el que funciona el propio microprocesado[r] central del ordenador y, por lo tanto, se puede decir que aprender a programa[r] en él es prácticamente lo mismo que aprender a programar el propio chip.

En el presente artículo se inicia lo que será el primer capítulo de un curso de ensamblador. Antes de nada, explicar que el ensamblador es el nombre genérico que se le da al conjunto de instrucciones que tiene programadas un microprocesador en forma de hardware y son las que realmente permiten sacarle utilidad al chip y hacer que éste sume, multiplique, mueva datos, realice comparaciones de elementos, etc.

Para simplificar las explicaciones y para que sea más fácil de entender, este curso se va a centrar en el estudio del ensamblador del microprocesador 8086 de la familia Intel, el que se instalaba en los primeros PC's que aparecieron, y pese a los años que tiene ya, todos los chips de última generación que se montan actualmente en todos los ordenadores PC más modernos, como los Pentium, Pentium MMX y Pentium II, siguen siendo compatibles en cuanto a funcionamiento binario con este chip (pese a tener muchas más funciones incluidas), y de ahí viene que los actuales ordenadores sigan siendo capaces de ejecutar los programas escritos para los primeros PC's sin ningún problema.

Veremos el ensamblador del procesador 8086 de la familia Intel

Pero antes de entrar en más detalles del funcionamiento interno del chip se van a detallar los componentes hardware indispensables que incorpora cualquier ordenador para que el chip pueda funcionar, así como su disposición de interconexión.

HARDWARE BASICO DE UN PC

Como se ha dicho, el microprocesador no es autosuficiente y requiere de toda una serie de elementos hardware y de conexiones físicas. Entre ellos que la complementen para que pueda llegar a funcionar.

A continuación se describen los componentes hardware indispensables:

1- Microprocesador: También se le conoce mucho por el nombre de C.P.U., siglas que provienen del inglés Control Process Unit (Unidad de Control de Proceso) y como el lector quizás ya sabe, no es más que un enorme entramado de microcircuitos que poseen unas dimensiones microscópicas (las C.P.U.'s se fabrican a escala de micras).
La función de la C.P.U., por explicarlo de forma resumida, es la de ir recibiendo instrucciones ensamblador (llamadas también código máquina) codificadas en formato binario con las operaciones que queremos que éste ejecute en cada momento y la de ir devolviendo los resultados o realizar las operaciones pertinentes.

2- Oscilador: Todo ordenador posee instalado un oscilador, un reloj que funciona a un ritmo medido en megahertzios (Mhz) y que será la velocidad a la que funcionará todo el sistema. Según sea el ritmo del oscilador, el ordenador se considerará más o menos rápido. Cuando apareció [el] primer modelo de 8086, estos ordenadores funcionaban a velocidade[s] de unos 4.77 Mhz. En la actualidad, cualquier ordenador de bajo coste funciona a 166 ó 200 Mh[z] ya están empezando a aparecer los primeros chip[s] que consiguen funcionar [a] 1000 Mhz (1 Ghz).
Para entenderlo, el oscila[dor] es en el ordenador como [el] ritmo del tambor en una galera de remos. Cada ve[z] que se produce una señal [del] mismo, el sistema realiza [un] ciclo de funcionamiento, enviándose información d[e] un sitio a otro, leyéndose [o] escribiéndose algo desde [o] en memoria y procesando [el] ciclo de la instrucción que esté ejecutando el microprocesador en esos momentos.

3- Módulos de memoria: La memoria no es nada más que chips en los que se almacenan tanto los datos como los propios program[as] ensamblador (código máquina, las instrucciones ensamblador) para que la C.P.U. puede ir leyéndolo[s]. Los módulos de memoria son grandes acumulaciones de celdas de memoria, ca[da] una dividida en ocho elementos llamados bits (cada uno de ellos puede contener el valor 0 ó 1).

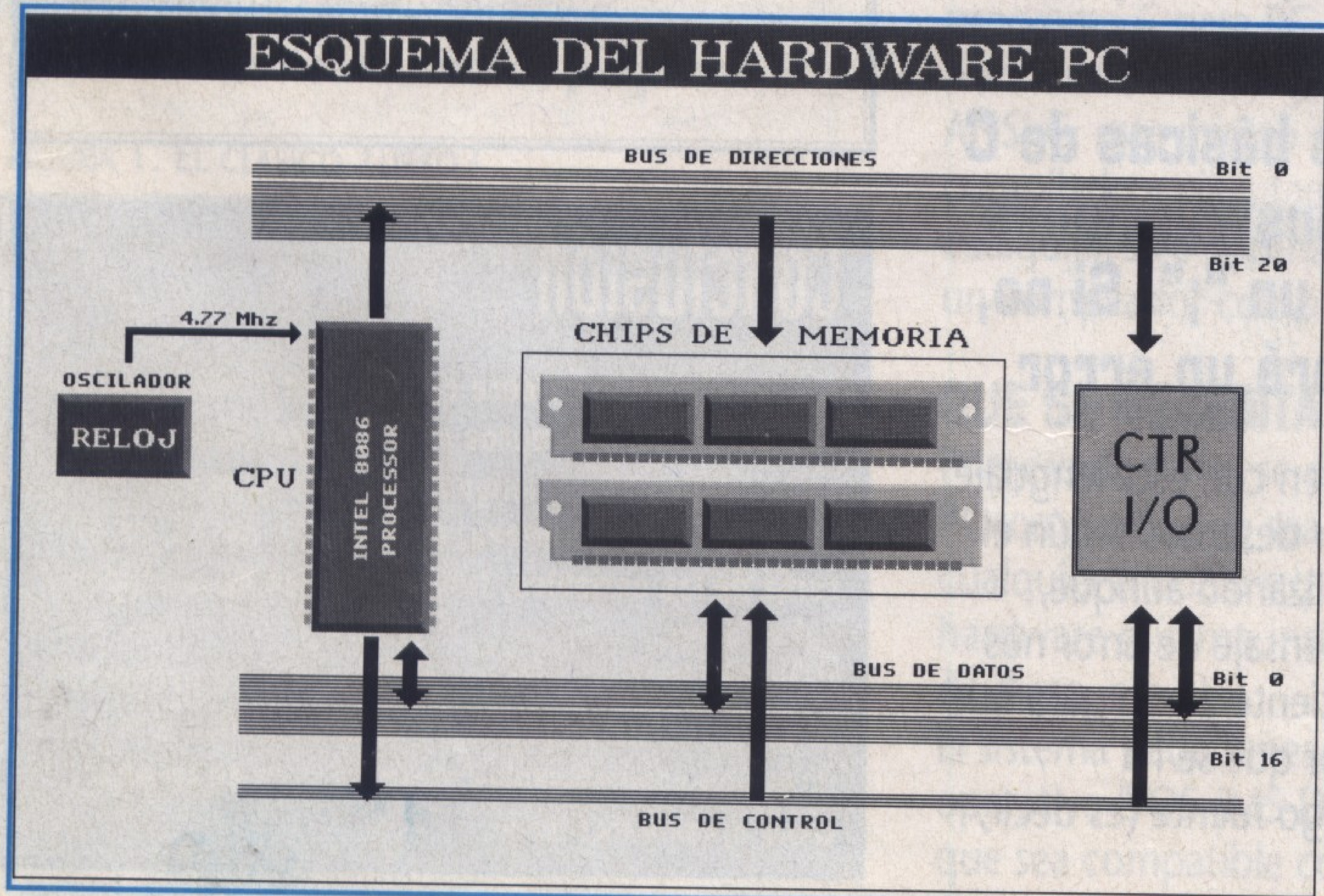


FIGURA 1. HARDWARE BASICO EN EL PC Y SU INTERCONEXION FISICA.

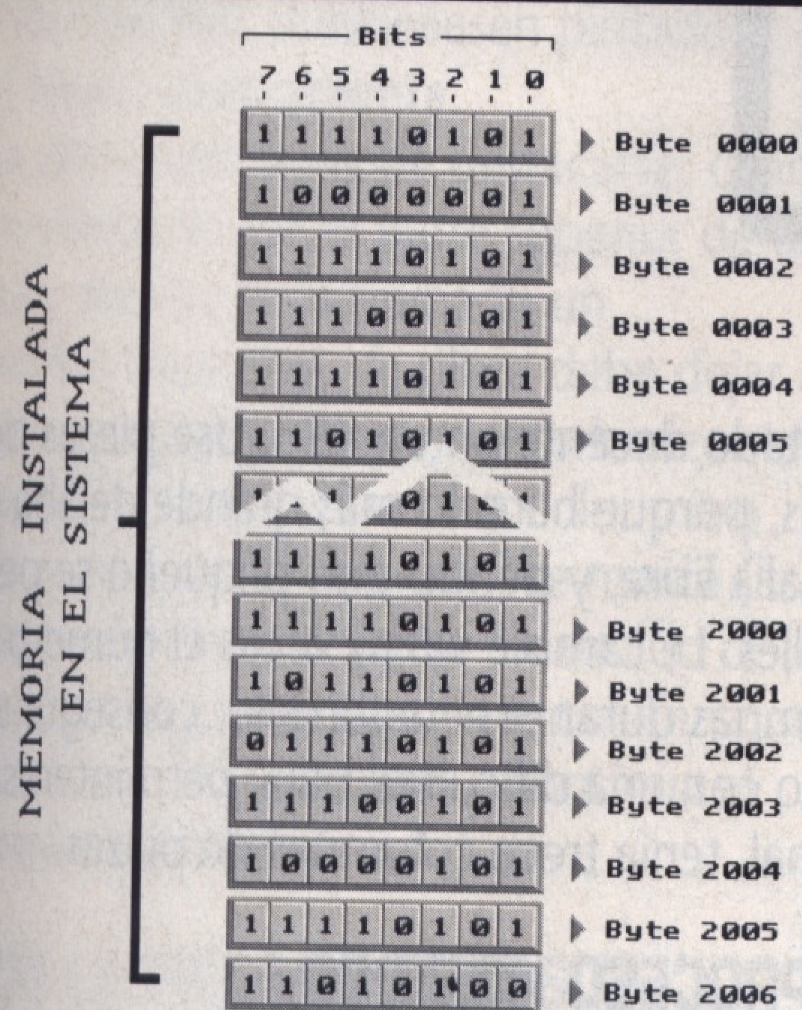


FIGURA 2. ESQUEMA DE MEMORIA DE UN ORDENADOR.

Para que la C.P.U. pueda indicar al sistema qué celdas de memoria quiere leer o escribir en cada momento, cada una de ellas posee asociado un número, la llamada *dirección física*, de forma que toda la memoria instalada posee las celdas numeradas secuencialmente, empezando por el cero y hasta el límite del que se disponga.

Las tablas de medidas de memoria en informática y en programación en general son muy importantes de saber y recordar siempre. A continuación se detalla:

A cada octeto de bits (cada celda de memoria) se le llama *byte*. Cuando se tienen 1024 bytes, se dice que se tiene 1 Kilobyte. Cuando se tienen 1024 Kilobytes se dice que se tiene 1 Megabyte de memoria. Cuando se tienen 1024 Megabytes, se dice que son 1 Gigabyte de memoria, y así sucesivamente (kilo, mega, giga, tera-, etc.).

4- Bus de datos: El bus de datos es el *cable* por decirlo así que conecta a la C.P.U. con la memoria y otros lugares. Por él pasan los bits de datos. Así por ejemplo, cuando el microprocesador envía el resultado de una operación, éste lo coloca en el bus de datos, por donde circulará hasta el módulo de memoria donde esté la celda donde se quiere almacenar el dato.

FIGURA 3. LOS CHIPS PARA PC SON COMPATIBLES CON EL 8086 QUE ESTAMOS ESTUDIANDO.



Nos será muy útil conocer la estructura del cable por donde circula la información binaria

5- Bus de direcciones: Como se ha dicho, cuando la C.P.U. envía o lee un dato hacia o desde la memoria a través del bus de datos, el sistema debe saber en qué dirección de memoria (recordar que cada celda tiene un número de posición asociado) se está refiriendo. Así, si la C.P.U. ha hecho una suma de dos valores y quiere almacenar el resultado en la posición de memoria número 60.000 (el octeto de bits nº 60000), el chip tendrá que mandar el dato por el bus de datos y la dirección a la que lo quiere mandar en el bus de direcciones, de forma que el sistema pueda dirigirlo hacia el lugar correcto (a la celda de memoria del chip de memoria correspondiente).

6- Bus de control: Este bus es el encargado de controlar qué clase de información binaria pasa por el bus de datos en todo momento y sincroniza en general el funcionamiento de la circuitería del sistema.

ANATOMIA DE UN BUS

Los buses son unos cables por donde circula información binaria (claro está), pero su anatomía es como sigue:

Cada bus se compone de un número X de pequeños cables situados en paralelo. Por cada uno de estos pequeños filamentos puede pasar en un momento dado 1 bit de información (0 si hay corriente ó 1 si no la hay) y el número de filamentos paralelos que posee un bus determina lo que conocemos por el *ancho de banda* de un bus.

El ensamblador no es otra cosa que el conjunto de instrucciones que tiene programadas un procesador en forma de hardware

Cuanto mayor sea el ancho de banda del bus de datos del ordenador, mayor es la cantidad de información que puede circular por éste simultáneamente por ciclo de oscilador, y por lo tanto mayor será su potencia de cálculo.

TABLA1. MICROPROCESADORES INTEL APARECIDOS EN EL MERCADO POR ORDEN CRONOLOGICO

- 3000 , 2N bits. Fabricado en TTL-Schottky.
- 4004 , 4 bits (usado en calculadoras).
- 4040 , 4 bits.
- 8008 , 8 bits. Fabricado en PMOS.
- 8021 , 8 bits. Fabricado en NMOS.
- 8035 , 8 bits.
- 8039 , 8 bits.
- 8041 , 8 bits.
- 8048 , 8 bits.
- 8049 , 8 bits.
- 8080 , 8 bits. Fabricado en NMOS.
- 8080A , 8 bits. Fabricado en NMOS.
- 8085 , 8 bits , hasta 3 Mhz. Fabricado en NMOS.
- 8086 , 16 bits , hasta 12 Mhz. Fabricado en NMOS.
- 8741 , 8 bits (incluye REPROM).
- 8748 , 8 bits (incluye REPROM).
- 8085A-2 , 8 bits , 5 Mhz. Fabricado en NMOS.
- 8088 , 16 bits , llega hasta 12 Mhz. (similar al 8086).
- 8087 , 16 bits (coprocesador 8086).
- 80186 , 16 bits , hasta 16 Mhz.
- 80286 , 16 bits , hasta 20 Mhz.
- 80287 , 16 bits. Coprocesador del 80286.
- 80386 , 32 bits , hasta 40 Mhz. Modelos SX,DX.
- 80387 , 32 bits (coprocesador del 80386).
- 80486 ,32 bits ,máx.100 Mhz. Modelos DX,DX2,DX4,SX y SL.
- Pentium , 32 bits , hasta 200 Mhz. Fabricado en BiCMOS.
- Pentium Overdrive (Ampliación para placas 486 a Pentium).
- Pentium Pro , 32 bits , llega hasta 200 Mhz.
- Pentium MMX, 32 bits, llega a los 233 Mhz.
- Pentium II, el chip más actual, llega a los 400 Mhz, aunque próximamente llegará a los 700 u 800.

Para decir un ejemplo, tenemos que el 8086 posee un bus de datos que posee un ancho de banda de 16 bits, y de ahí que a los ordenadores basados en él se los conozca popularmente como ordenadores de 16 bits todavía en la actualidad.

El ancho de banda de los ordenadores es un aspecto de su diseño que no ha cambiado mucho en estos años y actualmente todos los ordenadores son de 32 bits, aunque ya se están diseñando los primeros prototipos de PC que serán totalmente de 64 bits y que aparecerán entre 1999 y el 2000.

Autor: Roberto Sel
rselva@Tande.c

El sabor de un clásico

Este que ahora veremos fue mi primer juego, lo hice al mismo tiempo que terminaba de leer el manual de instrucciones del DIV, de forma que no aproveché todas sus características, y descubrí algunas de ellas cuando las necesité para continuar.

Es fundamental para empezar a trabajar escribir en papel lo que queremos que suceda, las interacciones entre los distintos elementos presentes en la pantalla. Mi idea inicial no fue exactamente igual al resultado, aunque sí resultó parecida. La mayor parte de añadidos se hicieron tras probar concienzudamente las versiones preliminares.
El proceso principal se encarga de iniciar el menú, que a su vez está compuesto de una serie de procesos destinados a informar al usuario, presentarle las opciones y la tabla con las mejores marcas. Una vez que el menú ha finalizado, al seleccionar el usuario la opción de jugar, el proceso principal entra en un bucle, dentro del cual se encarga de poner en marcha el proceso cuadro (básicamente, la pieza que cae), esperar hasta que se sitúa, y comprobar si esto afecta o no a los procesos estáticos que estaban depositados previamente, mediante el uso de unos procesos especiales que en el código llamo <*compro*>. Si hay una eliminación y algunas piezas caen, la comprobación ha de hacerse de nuevo. Cuando no hay más movimientos entre estos procesos, volvemos a lanzar otro «cuadro».
Este bucle sólo acaba cuando se supera el número de puntos que corresponde a cada fase, o bien cuando la pieza depositada por el jugador supera la altura permitida (esto no depende de la fase). Caso de que se haya superado el reto, el cambio de fase supone variar el número de colores que intervienen, la velocidad a la que bajan las piezas, y el número de puntos a superar. También cambia fondos y da un respiro al jugador (aunque siempre puede pulsar la tecla de pausa ya que este tipo de juegos debe tener pausas entre fases).

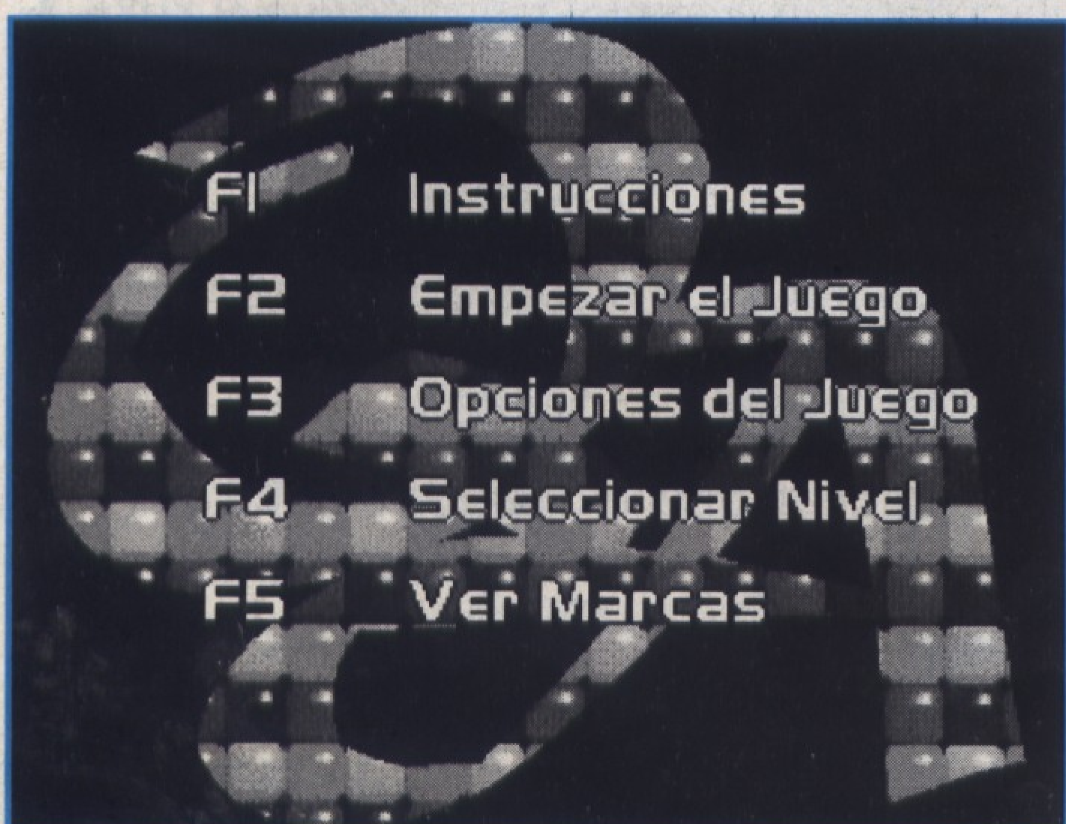
TAMAÑOS Y VELOCIDADES

Quisiera que os fijaseis en la resolución y el tamaño de las piezas en juego, así como en la velocidad del movimiento. Decidir entre varios tamaños no es tarea fácil y debe hacerse pronto. Usar un modo X de 320x240 me permitía mantener proporciones con modos superiores y permitir que el juego se viese en equipos con una vieja VGA. Entre los ordenadores entre los que probé se encontraban varios 386 y algún 486 en los que no se puede acceder al modo 640 por 480.
Pasé bastante tiempo haciendo pruebas con un *cuadradote* con movimiento elemental hasta decidir que una velocidad lateral de dos pixels y 50 imágenes por segundo era una buena combinación. Evidentemente, podía usar cuatro pixels a 25 imágenes, pero pensé que ya se ocuparía DIV de adaptarlo a la velocidad usando *set_fps*(50,3). El resultado es desesperante en un 386, aunque da un resultado decente en un 486. Usé piezas de 1
pixels, porque hacerlo más grande dejaba po
pantalla libre, y siendo más pequeño se perd
detalles. Le cambié varias veces el número de columnas durante el desarrollo, conseguí un juego con una dificultad baja, pero interesant
Al final, tenía trece columnas de piezas.

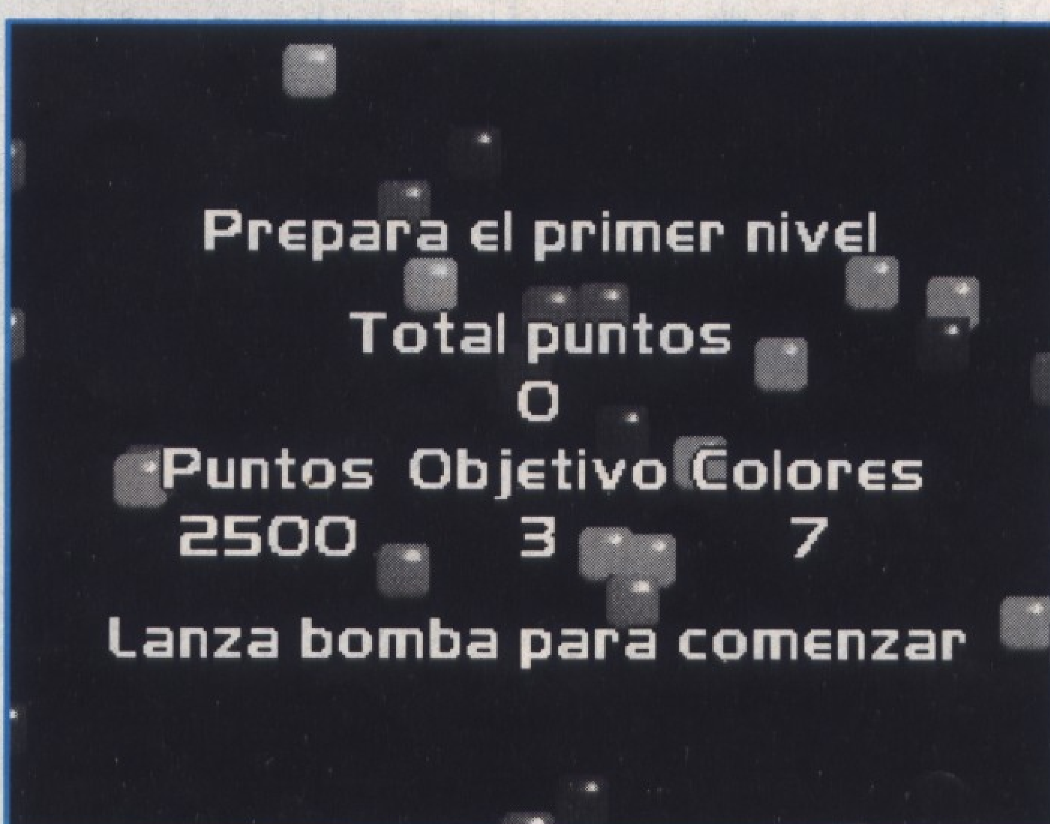
EL PROCESO «CUADRO»

Este es el proceso principal del juego, pues controla la pieza que cae. Bueno, en realidad sólo controla la primera de las tres partes de la pieza, ya que las otras dos son procesos de tipo *simil* que se limitan a moverse después de los demás, manteniendo su posición relativa respecto a su «padre». Con estos mismos procesos, y el proceso "otro", nos presenta la pieza que va a aparecer a continuación, cuando la que tenemos entre manos toque el fondo. Los colores de esta pieza se almacena en el vector <*prepara*>.
Una de las cosas más importantes del proceso *cuadro*, es que en realidad debería moverse de una columna a otra, ya que nunca puede depositarse en una posición intermedia. Sin embargo, su movimiento se realiza suavemen
Para lograr esto, utilizo una variable (*mov*) que almacena su velocidad. Si no se mueve, ésta e
0, y si pulsamos una tecla (izquierda o derech
vale *1* o *-1*. Por último, si se sitúa en la vertical de una columna, vuelve a valer *0*. En cada imagen, se multiplica por dos y se suma a la x.
Esta variable, una vez que toma un valor no nulo, sólo vuelve a valer *0* en el paso sobre las columnas ((*x-ancini)mod tam vale cero*).
Otra de las comprobaciones que hace el proceso es si se pulsa la tecla de *rotar colores*, e
cuyo caso permuta los gráficos de sus hijos sim
y el suyo propio. Comprueba que al menos haya transcurrido un cierto tiempo poniendo a cero el «*timer*» y no permitiendo una rotación hasta que haya superado un cierto valor. Esto s
hace para que el efecto de la rotación sea observado claramente por el jugador, ya que s
es demasiado rápido deja de tener utilidad.
En todo momento, el proceso recoge su posición en la rejilla del juego, es decir, la columna (*pos*). Usando ésta y la altura, es sencillo conocer la libertad de movimiento qu
tiene, puesto que la altura de cada una de éstas se recoge en el vector <*ultimo*>. Podría haber usado la función <*collision*>, pero no estaba familiarizado con ella en el momento

MENU PRINCIPAL DEL TITULO.



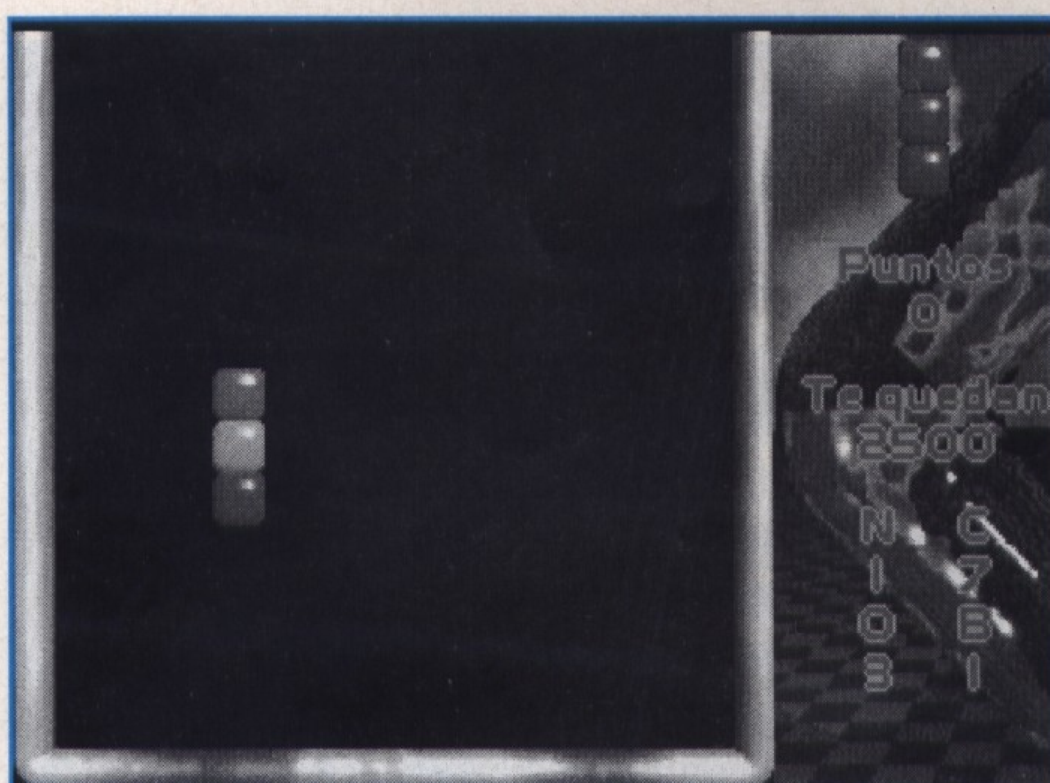
A PUNTO DE ENTRAR EN JUEGO.



hacer el juego. Además, este sistema valdría aunque no existiesen procesos en pantalla para dibujar los cuadros, si estuviesen pintados sobre el fondo directamente.
Hay una dificultad digna de destacarse: cuando entra en contacto con la parte superior de una columna y aún no ha terminado un movimiento horizontal, la pieza debe dejar de descender y seguir su movimiento horizontal, hasta decidir si se queda en esa columna o pasa a la contigua. En ese caso, se marca (a *1*) la variable <*excep*>, y se inicia un sonido de roce. Cuando llega a la siguiente columna, se devuelve el valor *0* a <*excep*> y continúa el movimiento habitual.

CUANDO LA PIEZA SE DEPOSITA

Más pronto que tarde, la pieza llega al fondo. En ese caso, su proceso «muere», pero antes almacena en la matriz <*fijos*> unos procesos de tipo «estatic» con los mismos gráficos y coordenadas que él y que sus hijos «simil», de forma que el jugador no observa ningún cambio. También cambia el valor de la variable global <*estado*>, de forma que el proceso principal sale de un bucle e inicia el proceso de comprobación de resultados. En el caso de que haya estado rozando, detiene el sonido y si nos hemos pasado de altura, lo señala en la variable <*final*>.
El proceso de comprobación de resultados es muy interesante. En realidad, el vector <fijos> es una matriz de dos dimensiones que almacena objetos de tipo «estatic», o en su caso un objeto (que representa el valor nulo) de tipo «falso». La comprobación se basa en encontrar el número de piezas de cierto color en contacto con las que se han situado recientemente (comprobar los contactos de las demás sería redundante). Para distinguirlas, las depositadas recientemente tienen un valor en la variable local «baja» de *-1*.
En un bucle del proceso principal se localizan estos procesos y se crea una variable llamada <*compro*> que tiene las mismas coordenadas que el objeto marcado. Esta variable sitúa su identidad en una variable similar a <*fijos*> llamada <*conexi*>, ocupando la misma posición que el objeto que marca, y procede a reproducirse a sí misma en las direcciones toleradas (*arriba*, *abajo*, *derecha* e *izquierda*, las diagonales no están permitidas). Sólo se reproduce en el caso de que las casillas adyacentes a la marcada tengan el mismo color, y no esté ya ocupada la casilla de <*conexi*> correspondiente. También, al nacer, incrementan una variable global (<*numcont*>), que acumula el total de <*compro*> que han surgido a partir de un determinado comienzo. Como el primer *frame* aparece después de la reproducción, antes de llegar a éste se han



UN SISTEMA SIMILAR A COLUMNS.

llenado todas las casillas posibles. Además, crea un proceso <*puntos*>, que será el encargado de señalar los puntos obtenidos por cada pieza, si no le pasa nada.
El control vuelve al proceso principal, que consulta la variable <*numcont*> para determinar si se procede a la destrucción o no. En el caso de que el número de conexiones no sea suficiente, se procede a señalar el lugar de comienzo poniendo la variable <*baja*> del <*compro*> a *1*. También aumenta su prioridad para asegurarse que se ejecuta antes. Después del *frame*, el proceso se dará cuenta que está marcado y destruirá su «puntos», y marcará a su descendencia. Para ello, repetirá el método de buscar en la matriz <*fijos*> colores adecuados, y modificará el <*baja*> y la prioridad de los objetos que ocupan la posición adecuada en <conexi>. En realidad, habría bastado asegurarse mediante las variables que mantienen las relaciones entre padres e hijos que todos sus descendientes quedasen marcados, pero en ese momento no se me ocurrió. Después, el proceso así marcado se suicida, tras devolver a la posición de <*conexi*> el valor <*idfalso*>.
Evidentemente, sus hijos procederán de igual forma cuando se den cuenta de la situación y marcarán a sus descendientes y se suicidarán. Esto, que aparentemente podría haber hecho con una señal <*s_kill_tree*>, es necesario por que la variable <*conexi*> debe quedar correctamente rellena con la identidad de <*idfalso*>. Supongo que internamente el funcionamiento de la señal antedicha debe ser muy similar, pero no sé como determinar las acciones que un proceso debe cumplir antes de morir.
En el caso en que los procesos no hayan quedado marcados tras el *frame* de su nacimiento, el número de contactos es suficientemente alto, por lo que se inicia el proceso de destrucción de los <*estatic*>. En primer lugar, iniciamos la animación de la explosión. En segundo, generamos unos objetos de tipo «restos» que simularán trocitos lanzados desde el cuadrado. Cuando la animación ha finalizado, se borra a sí mismo de la variable <*conexi*> (volviendo a ponerla a



LOS REFLEJOS SON IMPRESCINDIBLES.

<*idfalso*>), elimina al <*estatic*> correspondiente, y rebaja la altura de la columna correspondiente, copiando cada objeto que estuviese por encima a la posición en que debe quedar. Por cierto, que para que los <*estatic*> caigan suavemente, se indica en la variable <*baja*> de cada uno de ellos cuánto deben descender, y se les descongela para que puedan bajar. Hay que tener cuidado, ya que si tuviesen asociado un <*compro*>, también habría que hacerlo descender.
Además de todo lo anterior, hay una variable global (llamada <*contador*>) que almacena el total de <*compro*> activos, más el número de <*estatic*> en movimiento de descenso. Por eso

Los sonidos

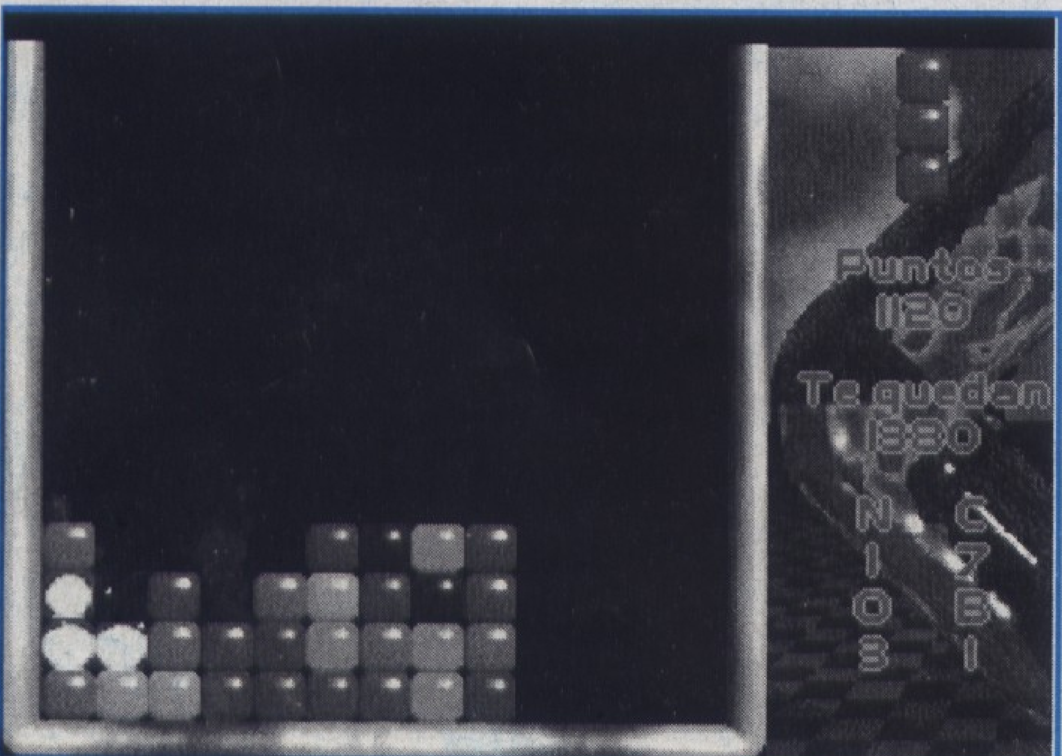
Es importante que los sonidos suenen en el momento adecuado, y también que se silencien (los sonidos continuos, especialmente) cuando dejan de tener sentido. Esto significa que los sonidos deben empezar a emitirse en el mismo frame, no inmediatamente junto al evento que los genera. Por ejemplo, si observamos la producción del sonido «caida», en realidad es el proceso principal el que lo genera (ver el bucle donde se crean los <compro>), en el caso en que la variable <prepara_golpe> contenga el valor <true>. Si fuesen los procesos <cuadro> o <estatic> los que lo produjesen, se podrían producir varios simultáneamente, en el mismo frame, podrían saturar el número de canales, o deformar el sonido. De esta forma, se centraliza su producción. Podría haber tenido en cuenta el número de piezas que hubiese caído en ese frame para dar distintas intensidades al sonido, pero eso es más complejo. En este juego no hay más sucesos que puedan coincidir, de forma que no hay que tener más precauciones especiales, y todos los demás sonidos se producen allí donde se observa la característica adecuada.
Los sonidos continuos, como el de roce, se inician en un determinado momento, y deben continuar en tanto se mantiene la circunstancia que los ha producido. Conviene, por tanto, dedicar una variable (en el juego <roce_suena>) para saber si está activo o no. En el caso que nos ocupa, bastaría haber usado la variable privada <excep>, ya que es la que indica que está rozando la pieza, y sólo hay un proceso «cuadro», que es el que puede o no rozar y por tanto quien activa o desactiva el sonido.

Tabla de marcas, reinventando la rueda

Desde un principio, pensé en anotar las mejores puntuaciones. Sin embargo, me tropezaba con el inconveniente de no disponer, en el lenguaje de DIV, de herramientas potentes que permitiesen al usuario introducir nombres o palabras, y almacenarlos en el disco.
¿Cómo podemos entonces hacer una tabla de marcas? usando un número para representar una letra, por supuesto. Decidí usar sólo seis letras por nombre, hice una variable con quince de estos elementos, y lo guardé en el disco. Desde ese momento, se carga al comienzo del juego, y cuando acabas una partida, si te corresponde meter una marca, te solicita que modifiques el nombre, salvándolo a continuación. Presentarlo era algo más complejo, pues no caben simultáneamente en pantalla, así que hay que desplazarlos usando (cómo no) las mismas teclas del juego.
Si se lee detenidamente el código del juego, se observa que cada nombre de la tabla de marcas está controlado por un proceso, que se sitúa relativamente a los demás, y controla la posición de la puntuación, y que mantiene un hijo por cada letra del nombre. Estas letras son posicionadas por los procesos hijo, que obtienen su posición del padre.
Después de concluir el juego, he visto otros con tablas de marcas mucho más depuradas que la mía, y he aprendido mucho. Por ejemplo, podría haber introducido los nombres desde el teclado, o indicar en pantalla las teclas a usar. Para acabar de estar satisfecho, la (sencilla) rutina de ordenación debe tener un error, y en ocasiones quedan desordenadas las marcas.

hay que actualizarla cuidadosamente antes de que el *compro* desaparezca. Y por último, alterar la variable <*baja*> del objeto <*puntos*> que ha creado, que a su vez habrá esperado pacientemente a que esto ocurra para mostrar los puntos y hacerlos subir por la pantalla. También podría habérmelo ahorrado creando este proceso en este momento, pero no me di cuenta.
Esta variable <*contador*> es la que le indica al proceso principal si ya han estallado todos y también han bajado los <*estatic*>a cerrar los huecos creados. Los objetos que han caído habrán llenado su variable <*baja*> con un *-1*, para que se pueda retomar el proceso con ellos, y así el programa procede hasta que no encuentra ni una sola cadena completa que destruir.
Lo más importante que destacaría de todo este sistema, es el papel tan importante que juegan las variables <*baja*> de cada tipo de procesos, que sirven a todos ellos para comunicarse y coordinarse, mediante el truco de alterar sus valores uno al otro. Esta forma de programar es muy delicada, ya que al usar una misma variable para varios cometidos, es fácil despistarse y provocar un efecto lateral difícil de localizar y depurar. Sin embargo, es un sistema que ahorra mucha memoria si se hace bien.

LA INFORMACION APARECE A LA DERECHA.



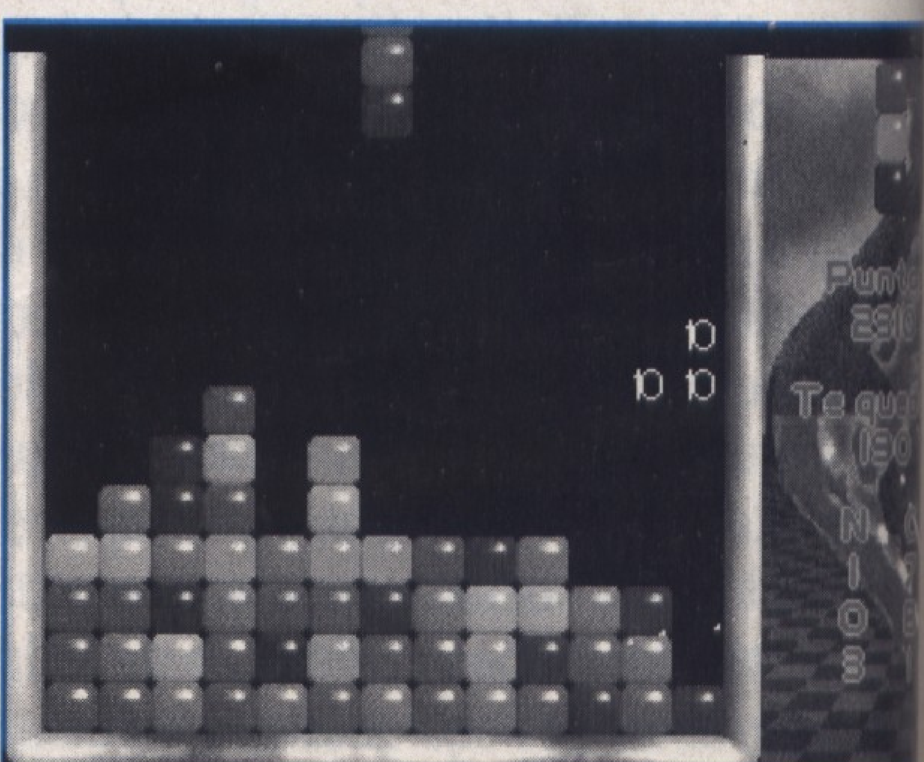
EL LANZAMIENTO DE LA BOMBA

La bomba no fue pensada inicialmente, fue una sugerencia de las personas que probaron el juego en una versión inicial. En realidad, hay una variable (<*estado_bomba*>) que controla si a lo largo de la caída anterior se ha pulsado la tecla correspondiente. El proceso encargado de manipular esta variable es <*cuadro*> (cuánto trabajo le he dado) y, en caso necesario, cambia el aspecto del proceso <*otro*>, que habitualmente muestra la pieza que va a caer a continuación, por una bomba, que tiene por gráfico una sencilla animación.
Cuando el programa llega a lanzar la siguiente pieza, genera un proceso «*obj_bomb*» en lugar de un proceso «*cuadro*», que tiene un comportamiento similar al del objeto «cuadro», pero más sencillo. La principal diferencia es que en cada dos *frame* cambia el gráfico, para que se vea la animación. Y cuando llega al fondo, no muere generando un <*estatic*>, si no se deposita a sí mismo en la matriz <*fijos*>, esperando ser destruida mientras continúa cambiando su gráfico.
Por su parte, el proceso principal, se encarga de destruir a la propia bomba (colocándole un compro) y a todos los <*estatic*> que tengan el mismo color que aquel que está inmediatamente debajo de esta (si ha caído en el suelo no destruye nada) por el mismo procedimiento. En la destrucción de la bomba usa un valor de cero (no incrementa la puntuación) y en la de los demás procesos, un valor 10. A partir de ahí, inicia las mismas comprobaciones que realiza cuando cae un «*cuadro*».

CAMBIO DE FASE

Cuando se alcanza el final de una fase, es decir, una cierta puntuación, algunas de las variables que controlan el desarrollo del juego han de cambiar. Cuando empecé a desarrollarlo, sólo diseñé una fase, pero después decidí qué elementos debían de cambiar de una a otra, agrupando todos e[...] en una estructura que llamé «*fase_var*». Ha[...] definida, al principio del juego, una matriz [...] los datos de todas las fases, de forma que basta cambiar alguno de los números para cambiar el aspecto, o los puntos que se ha[...] de superar, o las velocidades máximas y mínimas de caída en cada fase.
Si se llegase al final de la fase 16, que es la última diseñada, los datos vuelven a tomar[...] de la fase 1, pero incrementando la velocid[...] considerablemente. También he tenido en cuenta que una de las opciones del jugador [...] elegir la fase de comienzo.
Hay dos valores entre los registros de una fa[...] que contienen el menor color y el mayor co[...] que se va a ver en ésta, de forma que haya u[...] poco más de variación de una a otra. Aumen[...] el número de colores posibles incrementa notablemente la dificultad, así que conviene probar detenidamente todas las fases.
La velocidad de movimiento de la pieza que cae es otro factor muy relacionado con la dificultad. En realidad, parte de un valor mínimo que va aumentando paulatinament[...] tras cada lanzamiento hasta un máximo (qu[...] depende también de la fase). A partir de ah[...] vuelve otra vez al valor mínimo de forma brusca. Esto permite un respiro al jugador, [...] ha visto como cada vez tenía más difícil col[...] las fichas, y habrá amontonado varias cuand[...] vuelven a caer despacio.
El espacio entre fases se ocupa con una pequeña animación de la que son protagonistas las piezas que van a salir a continuación, que se desplazan por la panta[...] lentamente (a distintas velocidades), junto c[...] información complementaria. Esto da un respiro al jugador. En un principio, intenté q[...] los procesos «*test*» rebotasen unos contra o[...] al encontrarse, pero después de varias prueb[...] renuncié a ello, con lo que actualmente se ignoran pacíficamente. Al poco tiempo de desaparecer por un lateral vuelven a aparece[...] con la misma velocidad y dirección, pero es[...] vez por el contrario.

LAS COMBINACIONES PUNTUAN MAS.



EL TRA[...]
Cuando [...] mediana [...] construi[...] trabajo i[...] juego, q[...] herramie[...] Para los [...] usé un f[...] convirtié[...] necesari[...] usuario [...] Sharewa[...] imágene[...] calidad), [...] limité a [...] que usa [...] logrado [...] en cada [...] cambiad[...] Bastó de[...] correspo[...] coorden[...] tanteo, [...] en cuan[...] posición [...] Para los [...] manual [...] *www.ar*[...] a mano, [...] un *super*[...] ocupaba [...] (except[...] negro, y [...] «transpa[...] Tras ger[...] principi[...] pigmen[...] apliqué [...] Ahora t[...] límite a [...] primero [...] tenían, [...] un borr[...] cámara [...] un bue[...] mano. [...] que ocu[...] fondo, [...] colores [...] de caíd[...] el juego [...] de las b[...] fondos [...] No suc[...] cuando [...] fichero, [...] Los tipo[...] princip[...] incluía [...] especia[...]

EL TRABAJO GRAFICO

Cuando el programa ya funcionaba medianamente bien, es cuando me senté a construir gráficos más detallados. Al ser un trabajo importante en la construcción de todo juego, quisiera comentar básicamente las herramientas y la forma de usarlas.

Para los fondos de los laterales de cada fase, usé un fragmento de algunas fotos que tenía, convirtiéndolas a un tamaño aproximado al necesario con el Corel Photo-Paint v5 (soy usuario registrado). Después, con el programa Shareware Alchemy (sólo permite manejar imágenes a 640 x 480, pero posee gran calidad), les di el tamaño requerido, y las limité a 256 colores usando la misma paleta que usa *DIV* por defecto (se podría haber logrado un mejor efecto cambiando de paleta en cada fase, pero tendríamos que haber cambiado el gráfico de los cuadrados, etc.).

Bastó después usar *DIV* para importarlas al correspondiente *FPG*. Para encontrar las coordenadas donde situarlo utilicé algo de tanteo, pues era tarde y estaba algo confuso en cuanto a número de columnas, y la posición de los demás gráficos.

Para los cuadrados, usé el *POV-Ray* (hay un manual traducido al castellano en la dirección *www.arrakis.es/~pcostas*). En lugar de dibujarlos a mano, construí una imagen que sólo contenía un *superelipsoide*, moviendo la cámara hasta que ocupaba la práctica totalidad de la imagen (excepto las esquinas, claro). El fondo debía ser negro, ya que es el color que por defecto es «transparente» en la paleta predefinida de *DIV*. Tras generar unos cuantos, variando en un principio los colores y luego el valor del pigmento, hasta lograr diversos efectos, les apliqué un método similar al de los fondos.

Ahora tenía que hacer las barras que sirven de límite a los cuadrados. Esto sí costó, pues primero hube de calcular qué grosor relativo tenían, «construirlas» con *POV-Ray* (tras hacer un borrador a mano), y volver a jugar con la cámara para situarlas adecuadamente. Si eres un buen dibujante, quizá es mejor dibujarlas a mano. Después las corté en tres trozos para que ocupasen menos y las metí en el *FPG* del fondo, tras aplicarle un proceso de cambio de colores. Pensé en elegir un fondo para la «zona de caída» pero (tras varios intentos) decidí que el juego quedaba bien sin él. Acabé tan harto de las barras, que decidí que sólo haría cuatro fondos diferentes, aunque hubiese más niveles. No sucede lo mismo con los cuadrados, cuando tenía tiempo añadía uno o dos al fichero, pues la rutina era sencilla.

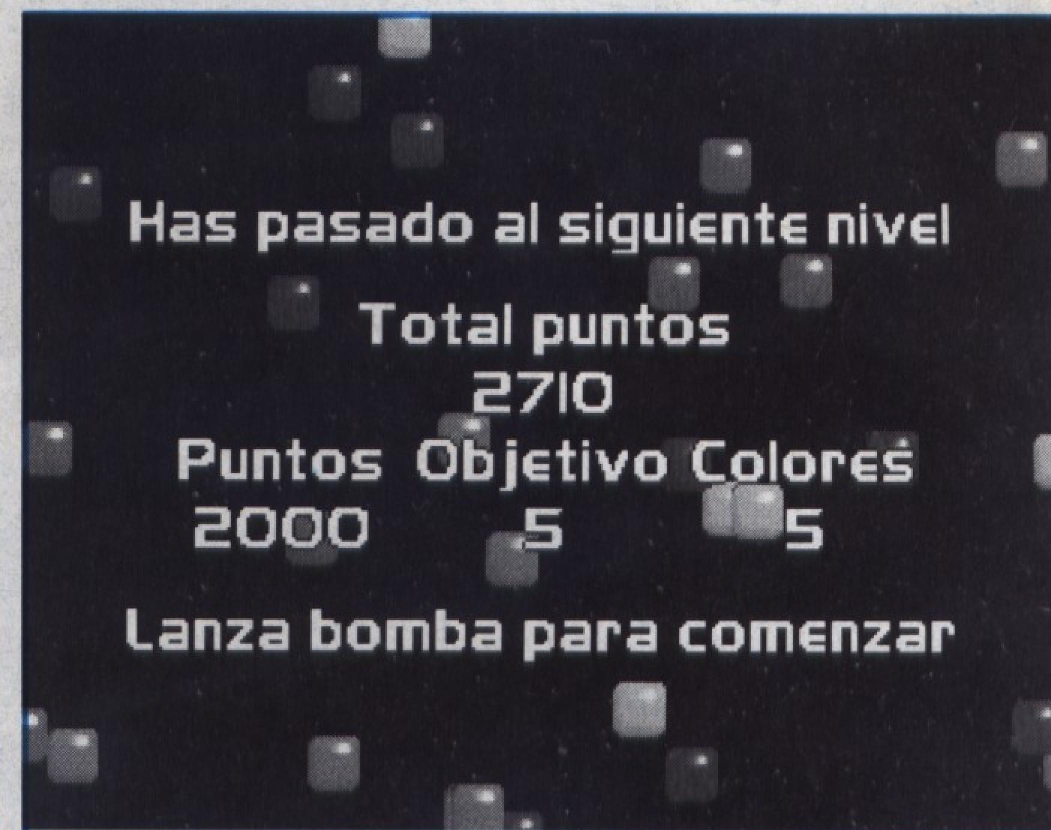
Los tipos de letra me marearon bastante. En un principio no me fijé en que algunos de los que incluía el DIV no llevan «ñ» y otros caracteres especiales (acentos y demás), con lo que aparecían muchos huecos. En un principio intenté prescindir de ellos, usando palabras sin «ñ» ni acentos, pero pronto descubrí que habría que cambiarlas.

La verdad es que fueron mis primeros experimentos con el generador de caracteres de DIV, y tampoco es que sea como para tirar cohetes. Ya sé que en esta resolución no se pueden hacer grandes alardes y pretender que el texto quede legible, pero debería haber usado algún pequeño reborde.

La animación de la bomba, también está hecha con POV-Ray, usando las facilidades de la versión 3.0 en cuanto a variación de parámetros. En poco tiempo tuve todas las imágenes. Convertirlas a la paleta de DIV fue coser y cantar, pero usar el DIV para colocarlas todas en un archivo FPG fue bastante más complicado. La verdad es que sería agradable que alguien con tiempo y conocimientos suficientes hiciese una herramienta para automatizar el proceso. No quiero ni pensar el trabajo de rutina que habría en un juego en el que cada personaje tenga un animación de 60 *frames*, y cada imagen tenga que ser leída, ordenada y almacenada desde DIV.

OPTIMIZACIONES

Cuando probé una versión del juego bastante avanzada sobre un 486 me llevé una tremenda decepción, ya que se arrastraba patéticamente (lo había programado sobre un Pentium 150). De forma que rebusqué en el manual del *DIV* instrucciones o métodos que me permitieran optimizarlo en velocidad sin reestructurarlo del todo. Lo primero que encontré fue permitir sólo un redibujado parcial a la pantalla. Al fin y al cabo, la mayor parte permanecía sin cambios todo el tiempo. Sin embargo, al colocar las instrucciones *<restore_type=partial_restore>* y *<dump_type=partial_dump>*, tuve la desagradable sorpresa de que los fondos no se repintaban cuando los cambiaba, de forma



que se superponían varias cosas. Por eso, cada vez que hay un cambio de fondos, vuelvo a cambiar el tipo de volcado de pantalla. Aún así, en máquinas lentas seguía sin repintar. Tras mucho pensar, llegué a la conclusión siguiente: como se salta varias *frames*, no redibuja el fondo cuando la pantalla está en volcado completo, ya que no pinta esas escenas, y cuando le toca pintar una escena, ya ha cambiado el modo de pintar, de forma que tampoco redibuja el fondo. Solución: después de cada cambio de redibujado parcial a completo, le obligo a esperar cuatro *frames* seguidas (en realidad, muy poco tiempo) para que pinte al menos una de ellas. Después, puedo volver al modo habitual sin problemas de fondo. De momento, no me ha fallado ni siquiera en un 386 SX.

Seguía con problemas de velocidad, de forma que decidí congelar a los *<estatic>* cuando no tenían que moverse. Sin embargo, los redibuja aunque no hayan cambiado el gráfico, y eso le lleva mucho tiempo. Si hubiese tenido más tiempo que dedicarle, incluso los habría suprimido, una vez que estuviesen quietos, dibujando su gráfico en el fondo, creándolos de nuevo cuando hiciesen falta para «bajar» un cuadrado por desaparición del inferior. Eso habría acelerado considerablemente el juego, ya que el número de procesos en pantalla tiene un impacto considerable en la velocidad.

Presentación, menu y opciones

No estoy satisfecho de lo que hice en este tema. En realidad, el juego usa pocas teclas para todo su control, y una de las opciones del jugador es cambiarlas (me di cuenta de lo necesario que podía ser al ver jugar a un zurdo con las teclas que yo había pensado). Sin embargo, el resto de opciones se controlan con esas teclas, y no hay un solo letrero que advierta qué se puede hacer con cada una de ellas.

En el aspecto de programación, el código fuente quedó bastante maltrecho al añadirle los menús, y los bucles de cambio de fase, etc.

Como no quería penalizar aún más la lectura de este proceso, y en un menú no es importante la velocidad, casi todo lo controla el proceso <menú>, mientras que el proceso principal lo llama y se sienta a esperar que haya terminado.

Este proceso <menú> carga un gráfico específico (el logotipo del fondo) y un sonido, inicia el proceso que baja este logotipo, manda a otro proceso a escribir las opciones en pantalla, y espera la pulsación de teclas del jugador. Tras cada pulsación, borra los textos y los reescribe. Cuando la opción escogida sea la de iniciar el juego, descarga gráfico y sonido, y devuelve el control al proceso principal. Éste elimina todos los procesos y borra la pantalla antes de continuar.

JUEGOS GANADORES: 2º EXPLOSS

Autor: Ismael Fernánde
ismafb@hotmail.c

Un programa explosivo

En este documento podremos encontrar comentada la forma en la que se ha hecho este juego, los trucos utilizados para ciertas cosas y cada proceso, detallado paso a paso, para que uno se dé cuenta de todo y logre comprender perfectamente todo el código fuente. La recomendación que hacemos desde aquí es tener delante no sólo este documento sino también el código fuente impreso para apreciar todos los detalles.

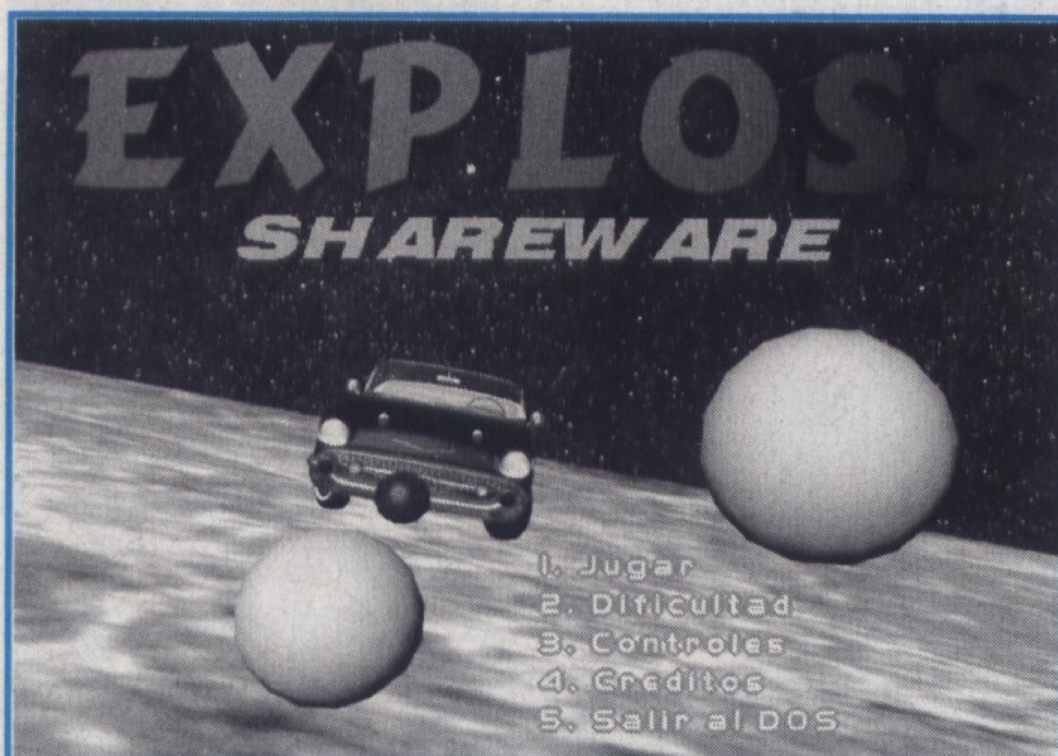
Para explicar todo el código fuente comenzaremos explicar, primero, cada proceso por separado; una vez hecho se pasa a comentar el programa principal. Está claro que en el código fuente primero aparece el programa principal y luego los procesos, pero consideramos que será más fácil entenderlo de esta manera.
Para cada proceso se escribe su interfaz, es decir, algo del estilo de:

PROCESS Nombre_Proceso (X,Y)

PROCESS COGER_BOMBAS (X,Y)

Este proceso es utilizado para visualizar en la pantalla el gráfico que si es cogido por el coche le proporciona cuatro bombas más; para ello, como parámetros recibe las coordenadas X e Y en donde el gráfico será visualizado, y como veremos en el programa principal, serán dos números aleatorios.
Asimismo, contiene una variable de tipo *PRIVATE*, es decir, cada vez que se llama a este proceso se genera para cada uno de ellos una variable que se llama *CON_1*. Lo primero que hace es poner el *flag EXISTE* a 1 para que ya no se generen nuevos, o sea, no puedan aparecer dos a la vez en pantalla. Elegimos el gráfico adecuado (en este caso el 91), y ponemos la coordenada Z a –1 para que se visualice por encima de todos los procesos.
Ahora, nos metemos en un bucle que irá de 0 a 100 a través de la variable *CON_1* que servirá como el límite máximo de tiempo que estará en pantalla este gráfico; dentro de este bucle lo que hacemos es comprobar si el coche está chocando con él. Si es así, incrementamos las bombas que teníamos en 4, comprobamos si nos hemos pasado del número máximo que puede tener el coche y con la sentencia *BREAK* salimos de bucle; justo antes de que acabe ponemos el *flag EXISTE* a 0 indicando que ya no hay nada en pantalla que proporcione cuatro bombas y así se pueda generar uno nuevo.

FIGURA 1.



PROCESS VIDAS( )

Éste se utiliza para visualizar el gráfico de coche como símbolo del número de vidas que se tienen, es decir, lo único que hace es poner el gráfico del coche en una posición a la izquierda y abajo para representar el número de vidas que nos quedan.
Lo primero que se hace es seleccionar el gráfico idóneo (30) y colocarlo en unas coordenadas X,Y y Z adecuadas (20,410,-1); posteriormente hay que introducirse en un bucle de tipo *LOOP ... END* para que no salga nunca de él.

PROCESS BOMBAS( )

Este proceso se usa para visualizar el gráfico de la bomba como símbolo del número de bombas que se tienen, es decir, es similar al anterior ya que lo único que hace es poner el gráfico de una bomba en la posición a la izquierda y abajo para representar al número de bombas que nos quedan.

PROCESS COCHE( )

Se utiliza para manejar el coche en función de las teclas que estén pulsadas o, en el caso de que se eligiera joystick, en función de lo que se apriete con él. En este proceso declaramos dos variables *INC_X e INC_Y*, inicializadas al valor 8, bien no van a ser utilizadas; se pueden poner como referencias para la distancia que se tiene que mover hacia los lados, hacia arriba y hacia abajo, y entonces bastará con cambiar el valor por otro simplemente en esta declaración para que se mueva más o menos en cada dirección.
Lo primero que se hace es elegir el primer gráfico del coche a visualizar (30) y las coordenadas iniciales (65,65,-1) y se mete en un bucle de tipo *LOOP ... END*, del que no saldrá nunca, a menos que destrocen al coche; dentro de este bucle lo que se hace es comprobar los intentos de mover los coches.
Para ello, lo primero es una sentencia *IF* (TECLADO) que se cumplirá si se eligieron las teclas como forma de controlar al coche; entonces, se harán comprobaciones de teclas de forma individual, es decir, se comprueba primero si se pulsa la tecla arriba y no se pulsa ni la de la izquierda ni la de la derecha y está en una coordenada Y > 65. En caso de que todo eso sea cierto, el gráfico del coche a visualizar es el coche mirando hacia arriba (3 se decrementa la coordenada Y en la distancia que se quiera mover al coche y se pone una nueva variable llamada dirección al valor de la dirección del coche; esta variable se utilizará por si acaso se dispara para saber en qué dirección tiene que ir el disparo. Las direcciones posibles son las siguientes:

ARRIBA ....
ABAJO ....
DERECHA ....
IZQUIERDA ....
ARRIBA + IZQUIERDA ....
ARRIBA + DERECHA ....
ABAJO + IZQUIERDA ....
ABAJO + DERECHA ....

A continuación se van comprobando todas las teclas para cada una de las direcciones, determinando, en cada caso, el gráfico adecuado para visualizar y moviéndolo la distancia adecuada en la dirección adecuada.
Llegado al punto en el que se comprueba qué ocurre al pulsar la tecla de la barra

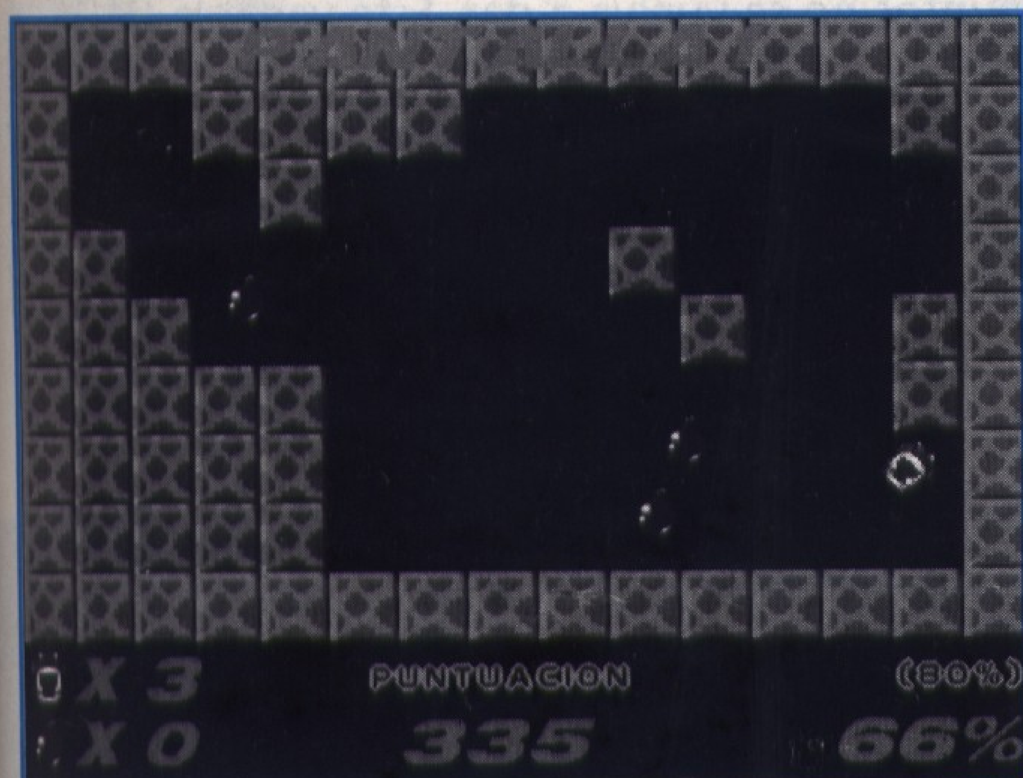


FIGURA 2.

espaciadora, lo que se hace es verificar si se está pulsando y se puede disparar, es decir, el coche no se encuentra subido en una baldosa marrón. En caso de que sea así, se comprueba otra variable, llamada *DISPARANDO*, para no poder dejar apretada la tecla para disparar. Si *DISPARANDO* vale 0, entonces crea un nuevo disparo a partir de las coordenadas X e Y del coche y en la *DIRECCION* que lleva el coche, después de crear el disparo pone la variable *DISPARANDO* a 1 para que así no se cree un nuevo disparo sin haber soltado la barra espaciadora; si no se tiene pulsada la tecla de la barra espaciadora, la variable *DISPARANDO* se pondrá a 0 para que se pueda disparar de nuevo cuando se apriete.

Después de comprobar si se intentaba disparar, se controla si se intenta poner una bomba con la tecla ALT; para ello, habrá que verificar si se pulsa esta tecla y hay bombas disponibles, es decir, *CUANTAS_BOMBAS > 0*. Si eso es verdad, se comprueba una variable (*PONIENDO_BOMBA*) que se utilizará para que no sitúe bombas sin haber soltado la tecla ALT, es decir, la variable *PONIENDO_BOMBA* realiza la misma función para las bombas que *DISPARANDO* para el disparo; en caso de que *PONIENDO_BOMBA* fuera 0 decrementa el número de bombas disponible, crea una bomba en las coordenadas X e Y del coche y pone *PONIENDO_BOMBA* a 1, si no se pulsa ALT esta variable es puesta a 0.

Luego viene la sentencia *ELSE* que corresponde al *IF* (TECLADO), es decir, a partir de ahora aparecen los controles del coche en caso de que el control elegido en lugar de teclado fuera el joystick, no merece la pena repetirlo porque es exacamente igual que con el teclado, pero con el joystick (mirar la estructura global joystick en el manual de referencia y dudas aclaradas); para el intento de disparo y de poner bombas con el joystick ocurre lo mismo que con el teclado.

Después, se hace una comprobación para ver si el coche está chocando con un obstáculo (baldosa marrón); en caso de que sea cierto coloca la variable *PUEDE_* a 0 para que el coche no pueda disparar cuando está encima de ellas; además, con la sentencia *FOR* que hay se consigue un retardo que permite que el coche vaya más lento. En caso negativo, coloca *PUEDE_DISPARAR* a 1 para que se pueda disparar (ya que no pisa la baldosa marrón).

Posteriormente, realiza una comprobación para ver si colisiona con alguna bola (grande, mediana o pequeña); en tal caso sitúa los gráficos de la explosión (80-85), pone la variable *MUERTO* a 1, decrementa el número de vidas y ejecuta la sentencia *BREAK* para que se salga del bucle *LOOP END* del inicio.

Process inicia_fondo (STAGE)

Se utiliza para iniciar los gráficos de las baldosas de ambos colores (verdes y marrones) por toda la pantalla, es decir, al empezar cada pantalla rellena toda la pantalla de baldosas verdes y marrones. El parámetro STAGE, que recibe siempre, será el número de pantalla a la que vamos a acceder; entonces, con este parámetro se ejecuta una sentencia SWITCH para iniciar la pantalla adecuada, es decir, poner los cuadrados verdes por toda la pantalla y los marrones en ciertas posiciones. Como el número de pantallas es de 10 habrá desde CASE 1 hasta CASE 10, aunque simplemente comentaremos CASE 1 y el CASE 2.

En el CASE 1 (primera pantalla), lo que se hace es rellenar toda la pantalla con bloques verdes (PROCESS FONDO (X,Y) que explicaremos más adelante). Para ello, se utilizan dos bucles FOR anidados con las variables CONT1 y CONT2, que variarán las coordenadas X e Y de las baldosas verdes para colocarlas por toda la pantalla; el bucle más externo variará la coordenada X de 43 en 43 unidades, que se corresponde con el ancho de cada baldosa verde (así quedarán unas al lado de otras), y el bucle interno variará la coordenada Y de 43 en 43 unidades, es decir, el ancho de cada baldosa verde (así quedarán unas debajo de otras). Como todas las pantallas tendrán mayoría de baldosas verdes, estos dos bucles, como se puede observar, se encontrarán en todos los CASE y así bastará solamente con iniciar luego los obstáculos que se quieran (baldosas marrones). Una vez que se ha iniciado la primera pantalla (toda con baldosas verdes) cuando se haya pasado se iniciará la pantalla 2 (CASE 2).

En el CASE 2 (segunda pantalla), lo que se hace es colocar cuatro obstáculos (baldosas marrones) en ciertas coordenadas (PROCESS OBSTACULO (X,Y) que serán comentadas más adelante), y luego iniciar por completo todo de baldosas verdes (con los dos bucles anidados que utilicé en el CASE 1). Puede parecer que en algunos sitios haya dos procesos superpuestos (una baldosa verde y una marrón), pero como veremos más adelante esto se evita por una comprobación de colisión en el proceso FONDO (X,Y).

El resto de sentencias CASE funcionan igual que estas primeras: la diferencia es que se inician procesos de tipo obstáculo en distintas coordenadas. Una vez iniciada la pantalla adecuada, este proceso finaliza ya que el hecho de que se sigan visualizándose en pantalla los gráficos se debe a los procesos FONDO y OBSTACULO que a continuación pasamos a detallar.

PROCESS INICIA_BORDE ( )

Lo que hace es visualizar el gráfico que forma el borde de la pantalla, que está formado por un grupo de bloques grises. Selecciona el gráfico (52) y las coordenadas en las que debe colocarse (321,241) y se entra en un bucle de tipo *LOOP ... END* en el que sólo ejecuta la sentencia *FRAME*. Así, cuando sea llamado por única vez el gráfico se estará visualizando siempre en esas coordenadas.

PROCESS OBSTACULO (X,Y)

Se usa para visualizar el gráfico de las baldosas marrones en las coordenas X e Y que se le indican como parámetros; selecciona el gráfico adecuado (50) y se entra en un bucle *LOOP ... END* en el que sólo se ejecuta la sentencia *FRAME*, es decir, no finaliza nunca a no ser porque otro proceso le mande la señal de morir.

PROCESS BOMBA (X,Y)

Se utiliza para visualizar el gráfico de una bomba en las coordenadas X e Y que se le indican como parámetros; lo que se hace es seleccionar el gráfico adecuado (90) y ejecutar un bucle *LOOP ... END* en el que sólo se ejecuta la sentencia *FRAME*, es decir, no finaliza nunca a no ser porque otro proceso le mande la señal de morir. Las coordenadas X e Y que recibirá serán siempre las coordenadas del coche en el momento en que se mandó poner la bomba.

PROCESS INICIO_BOLA_GRANDE (X,Y)

Éste es utilizado para visualizar los gráficos que indican que se va a iniciar una nueva bola; se trata de los gráficos del 70 al 74. Lo que se hace es ejecutar un bucle *FOR*, desde 1 hasta 3, y dentro de este un bucle *FROM* desde 70 hasta 74 para que los cuatro gráficos se visualicen uno detrás de otro durante tres veces. Una vez que finalizan estos bucles lo que se hace es iniciar una bola grande en sus mismas coordenadas.

PROCESS BOLA (X,Y)

Con este proceso se puede controlar a las bolas grandes. Se inicializa el gráfico adecuado (60) y se entra en un bucle *LOOP* en el que se comprueba si choca con un disparo; si es así destruye el disparo que chocó, incrementa los puntos y genera dos bolas medianas que se crean en sus mismas coordenadas y se mueven en distintas direcciones, a causa de *INC_X e INC_Y* ya que a la primera bola mediana se le pasan como parámetro con signo positivo y a la segunda con signo negativo. Además, ejecuta la sentencia *BREAK* para que se salga del bucle y desaparezca por tanto la bola grande. Se comprueba también si choca con una bomba,

Process disparo (x,y, dire)

Este proceso se usa para visualizar el gráfico del disparo en ciertas coordenadas y que se moverá en una dirección determinada que recibirá como tercer parámetro. Las coordenadas en las que se inicia serán las del coche en el momento de disparar, por eso, lo que se hace es seleccionar el gráfico (38) y entrar en un bucle LOOP del que sólo se saldrá si se sale de la pantalla o choca con un obstáculo. Lo primero que hace dentro del bucle es, en función de DIRE a través de un SWITCH, mover al disparo en cierta dirección (que será la misma que tenía el coche cuando se disparó). Una vez que se ha movido el disparo, se comprueba si éste se sale de la pantalla en cuyo caso solamente se ejecuta la sentencia BREAK para que se salga del bucle y el disparo que se salió de pantalla desaparezca. A continuación si comprueba con un obstáculo; en caso de ser cierto se toma el identificador del obstáculo con el que choca y se manda matar a ese proceso obstáculo, aparte de incrementar los puntos y el porcentaje y comprobar si el porcentaje es mayor que 80; si es así, la pantalla estará pasada (se pone PASADA a 1) y luego se ejecuta la sentencia BREAK para hacer desaparecer también al disparo que chocó con el obstáculo.

en caso de ser así incrementa los puntos y se hace desaparecer a la bomba que la mató, ejecutando *BREAK* para que se salga del bucle y desaparezca (ahora no se generan dos bolas medianas). Después de estas comprobaciones se varían las coordenadas X e Y en *INC_X* e *INC_Y*, respectivamente, y se comprueba si la bola choca con una pared, en cuyo caso cambia de signo *INC_X* si choca con una pared lateral, e *INC_Y* si choca con la superior o inferior.

PROGRAMA PRINCIPAL

Es la parte del programa encargada del control del juego; lo primero que hace es cargar los archivos necesarios, tanto los de sonido como los de gráficos y fuentes; además se inicia el modo gráfico a 640X480 y el número de imágenes por segundo a 200. Se inician dos variables del control de juego: *NIVEL*, que cada vez que se inicie el juego será puesto a 2 (nivel medio) y *TECLADO* como controlador del coche (es puesto a 1).

FIGURA 3.



Posteriormente, se entra en un bucle de tipo *LOOP* del que sólo se saldrá cuando se pulse la tecla 5 (salir al DOS); éste será el bucle general (para iniciar una nueva partida).

BUCLE GENERAL

Al entrar en él se inician variables al comienzo de cada partida, como los puntos, la pantalla actual, el número de vidas, de bombas, si está muerto, si se quiere acabar una partida, si se ha continuado alguna vez o si se ha pasado una pantalla; posteriormente, se pone el gráfico de presentación y se escriben las opciones, entrando en un nuevo bucle *LOOP* en el que se comprueba la tecla pulsada. Si se pulsa 1 se sale de este último bucle *LOOP* (se ejecuta *BREAK*) y se inicia una nueva partida, si se pulsa 2 se elige el nivel de dificultad, y en función de la dificultad que se seleccione se varía la variable *NIVEL* a 1, 2 o 3. Si se elige 3 se escoge la forma de controlar el coche (por teclado o joystick); si se elige teclado la variable *TECLADO* se pone a 1 y si se elige joystick se pone a 0. Si se elige 4 se pone la pantalla de créditos y si se pulsa el 5 se sale del juego con *EXIT*. A partir de aquí aparece todo el código acerca de cada partida en individual, es decir, se inicia un nuevo bucle, esta vez de tipo *REPEAT ... UNTIL* del que sólo se saldrá cuando no queden vidas (nueva partida).

BUCLE GENERAL POR PARTIDAS

Aquí se comprueba si se ha pasado una pantalla. Como se puso la variable *PASADA* a 1 se inicia una nueva pantalla, la primera, y lo que se inicia el borde la pantalla, el fondo y escribir los textos que indican la puntuación, las vidas, etc. Luego se comprueba si le han matado; en caso afirmativo, no se inicia el fondo (no se llena de baldosas desde el principio sino que se deja tal y como estaba cuando le mataron), eliminando todas las bolas que existían y todos los disparos y se vuelven a escribir los textos de vidas, bombas, puntos... Ahora se entra en el bucle que regula las pantallas, que es de tipo *LOOP*.

BUCLE GENERAL POR PANTALLAS

De este bucle sólo se sale si le matan, si se pasa de pantalla o si se pulsa ESC, además se puede comprobar si la puntuación es mayor o igual que *PROXIMA_VIDA* que indica a cuántos puntos se conseguirá la próxima vida; para ello, se verifica la puntuación con un margen de 10 puntos y se comprueba que no se ha dado la vida, en cuyo caso se da (incrementando el número de vidas y *PROXIMA_VIDA*), y se pone la variable *DADA* a 1 para que no se vuelva a dar; es decir, si se llevan 3005 puntos y se da por primera vez la vida, si por casualidad se está un rato sin hacer puntos no se vuelva a dar. Esta variable *DADA* sólo se pone a 0 cuando ya no se esté en el rango en que se puede dar. En este bucle se generan números aleatorios y se comprueba que si es menor que cierto valor se inicia una bola grande o un objeto que da cuatro bombas más; el rango en el que se sacan números aleatorios cada vez es menor a causa de las variables *PANTALLA* y *NIVEL*, que son utilizadas dividiendo para que el rango disminuya a medida que ambas aumentan de valor. Por otro lado, se comprueba si se pasa de pantalla o le matan o se pulsa Escape para salir de este bucle por pantallas.

FIN DEL BUCLE POR PANTALLAS

Si se salió del bucle porque nos habíamos pasado la pantalla, lo que hacemos es congela los procesos coche, bolas y disparos y escribe en pantlla «CLEAR», para indicar que se la ha pasado y a continuación muestra la foto «manga» en pantalla, en función de la pantalla que nos acabemos de pasar (cuando se llama a *PUT* se indica la pantalla en la que estamos, ya que las fotos están unas detrás de otras con los códigos del 1 al 10). Ésta se visualizará hasta que se pulse espacio o el boton 1 del joystick. Después se comprueba si la pantalla que se acaba de pasar es la 10, en cuyo caso se coloca el final del juego: el movimiento de texto por pantalla. Para dicho final se sitúa el gráfico de presentación y se inicializan los textos que, posteriormente, se moverán; por medio de la sentencia *MOVE_TEXT* se van moviendo hacia arriba hasta que se llega a cierta posición. Esto se hace con todos los textos hasta que acaban, se ejecuta *BREAK* para volver a la pantalla de presentación; si no se ha ejecutado es porque la pantalla no era la 10, por lo que se tendrá que incrementar la pantalla (en el caso de que se pasen de pantalla). Ahora llega la comprobación del número de vidas. Se verifica si el número de vidas es cero y no se ha continuado. Si ocurre esto se da la opción de continuar consistente en dos bucles. El externo irá de 9 a 0 contando (simulando el tiempo que queda para poder pulsar) y el interno irá de 0 a 10, provocando un retardo (para que no baje de 9 a 0 rápidamente) y se comprueba si se pulsa S o N. Si se pulsa S se ponen las vidas a [illegible] las bombas a 4, indicando que le han matado (*MUERTO*=1), para que no inicie la pantalla en la que estábamos desde el principio, y se pone *CONTINUADO* a 1 señalando que ya se ha continuado. Si se pulsa la N se ponen las vidas a –1, y si las vidas son mayores que 0 es porque se ha continuado, mientras que si son menores que cero se ponen a cero; entonces se matan a los procesos bomba, vidas y bombas y se borran los textos, llegando al final del bucle por partidas del que sólo se sale si las vidas son 0.

Autor: Antonio Marchal
hizo@100mbps.es

El juego del Cinquillo

La idea consiste en realizar un juego de cartas. De todos ellos se eligió uno muy sencillo, pero no por eso menos entretenido, el cinquillo. Se juega contra el ordenador y las rutinas de inteligencia, aunque simples, cumplen totalmente su cometido, como podréis comprobar.

A continuación se comentarán los puntos importantes que son necesarios para realizar el programa.

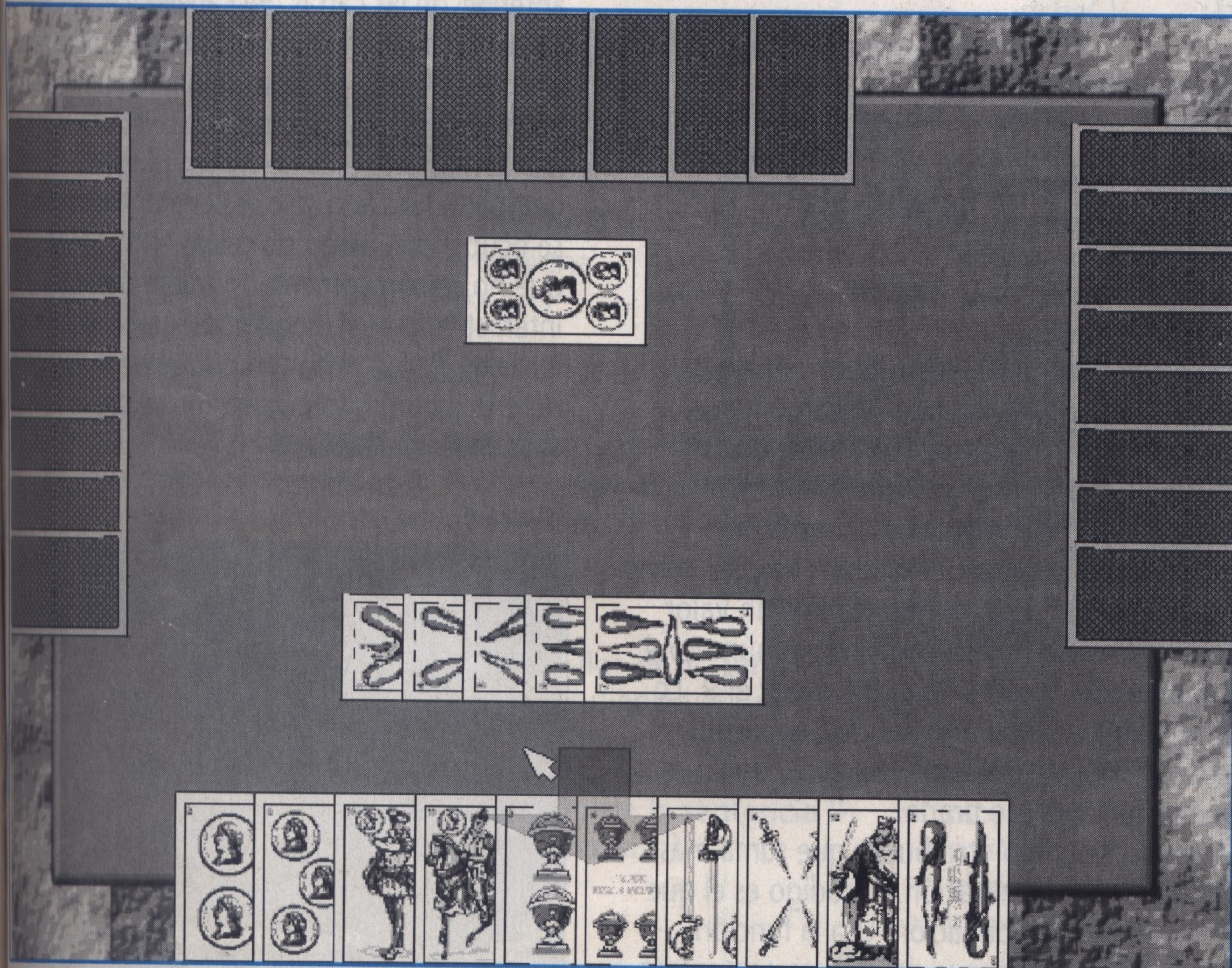
VARIABLES

```
cartasbaraja[40];
```

Esta variable guarda la baraja, es decir todas las cartas. Al comienzo de cada mano, se rellena con todas las cartas ordenadas.

```
STRUCT jugador[3];
  cartas[10];
  numerocartas;
END
```

En esta estructura se guardan los datos de las cartas de cada jugador.



UN CLASICO SOBRE EL TAPETE.

```
STRUCT palotapete[3];
  cartas[9];
  numerocartas;
END
```

Aquí se guardan las cartas que hay sobre la mesa.

```
turno=0;
finjuego=-1;
turnoinicial=0;
partidasganadas=0;
partidasperdidas=0;
```

Todas estas variables controlan aspectos generales del programa, como el turno, las partidas ganadas y perdidas, o si se ha acabado completamente el juego o no.

PROCESOS

Existen unos procesos esenciales, que manejan todo lo referente al control de cartas. Se les llama desde el programa principal y son los siguientes:

```
llenar_cartas();
barajar_cartas();
repartir_cartas();
pinta_cartas(valor);
```

Se encargan de todo lo relacionado con las cartas: barajarlas, repartirlas, pintarlas en pantalla, etc. Dentro del programa principal, la líneas de código son las siguientes:

```
turno=turnoinicial;
turnoinicial++;
IF (turnoinicial>3)
  turnoinicial=0;
END
maneja_turno();
```

El cinquillo, uno de los más conocidos juegos de cartas, también puede tener su lugar dentro de los PCs

Se encargan de manejar todo el control de turnos e interactividad con el jugador. Pero vamos a ver más detenidamente alguno de estos procesos. El primero es llenar_cartas(), que es un simple bucle que llena la baraja, en orden y por palos: oros, copas, espadas y bastos. El siguiente proceso, relacionado con las cartas, es barajar_cartas(). Para barajar, se va recorriendo toda la baraja y se intercambia la posición con cualquier otra carta de la baraja. Para intercambiar las cartas, se hace uso de otra variable:

```
pasavalor=cartasbaraja[0];
cartasbaraja[0]=cartasbaraja[valorcambio];
cartasbaraja[valorcambio]=pasavalor;
```

Primero se mueve la primera carta, luego las cartas 2 a 39; por último, la del final, la 40. Esto se hace de esta manera para evitar comprobaciones. Se ha elegido un método basado en la función rand(), pero que evita la carta que está seleccionada en ese momento para el intercambio.

Juegos ganadores: 3º Cinquillo

El siguiente proceso reseñable es ordena_cartas(cual), que ordena las cartas de la mano del jugador que le indiquemos. Esto es necesario para que el jugador en cuestión tenga las cartas ordenadas en todo momento. Para ordenarlo, lo que se hace es ir recorriendo toda la mano, poniendo al principio el valor menor. Cuando se llegue al final de la mano, todo estará ordenado. El código con los dos bucles, y el corazón de la rutina de ordenación, es el siguiente:

```
FROM contador1=0 to 8;
    posicionmenor=contador1;
    valormenor=jugador[cual].cartas[contador1];
    FOR (contador2=contador1;contador2<10;contador2++)
        IF (jugador[cual].cartas[contador2]<valormenor)
            posicionmenor=contador2;
            valormenor=jugador[cual].cartas[contador2];
        END
    END
    pasavalor=jugador[cual].cartas[contador1];
    jugador[cual].cartas[contador1]=jugador[cual].cartas[posicionmenor];
    jugador[cual].cartas[posicionmenor]=pasavalor;
END
END
```

De todos los procesos usados, el más complicado es maneja_turno() ya que, además de controlar toda la interfaz con el jugador y manejar los turnos, también se encarga de la inteligencia del ordenador. Lo primero que encontramos es una serie de variable PRIVATE, que es la siguiente:

```
posiblestiradas;
```

Controla las tiradas posibles, contándolas; otro contador de tiradas es ntiradas_cpu.

```
STRUCT tiradascpu[8]
    posicionmano;
    puntuacion;
END
```

Esta estructura guarda las jugadas posibles que puede hacer la máquina en cada turno.
Además del código que maneja el turno del jugador, el más interesante es el que maneja el turno del ordenador. En los dos casos, lo primero que se hace es comprobar si el jugador que tiene el turno puede tirar. Si no puede debe pasar el turno al siguiente jugador.

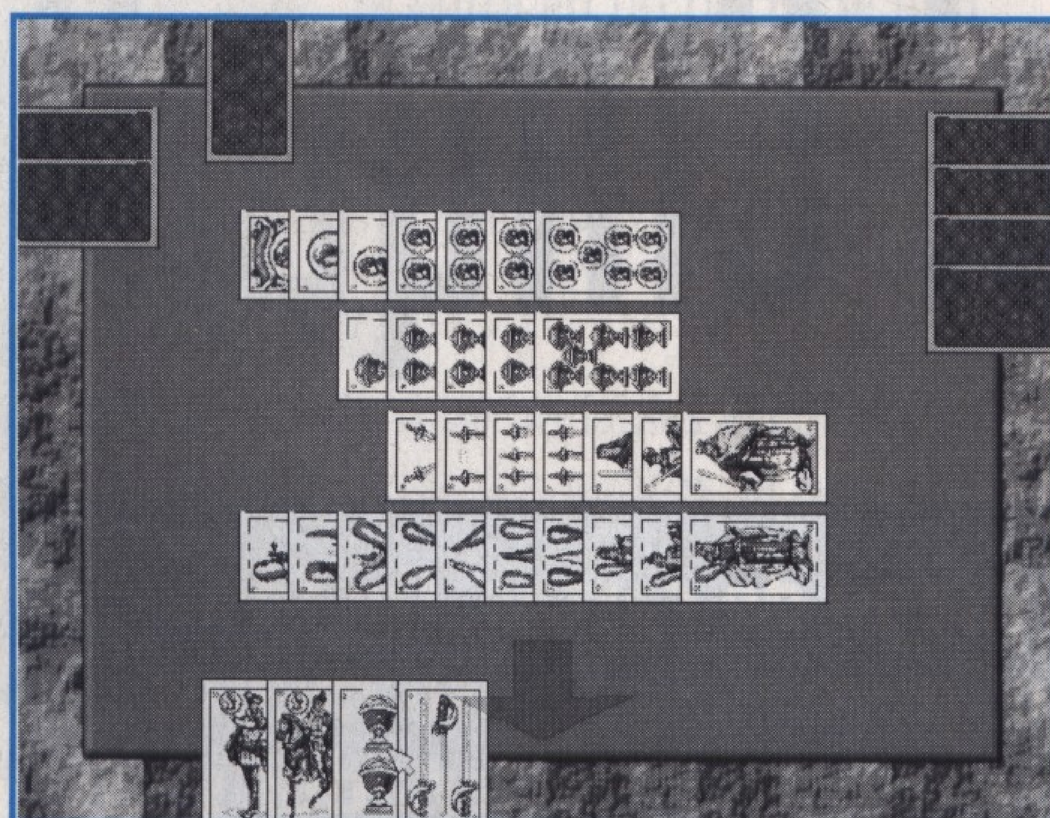


FIGURA 2.

Mediante un bucle y la función comprueba_tirar(), se comprueba este hecho, usando el siguiente código, que cuenta las posibles tiradas:

```
FOR (contador1=0;contador1<jugador[turno].numerocartas;contador1++)
    IF (comprueba_tirar(jugador[turno].cartas[contador1]))
        posiblestiradas++;
    END
END
FOR (contador1=0;contador1<jugador[turno].numerocartas;contador1++)
    IF (comprueba_tirar(jugador[turno].cartas[contador1]))
        tiradascpu[ntiradas_cpu].posicionmano=contador1;
        tiradascpu[ntiradas_cpu].
puntuacion=
            abs(4-(jugador[turno].cartas[contador1] % 10));
        ntiradas_cpu++;
    END
END
```

De estos dos bucles, el primero es para el turno del jugador, ya que únicamente se comprueba si se puede tirar. El segundo bucle es el primer paso del código de la inteligencia del ordenador. Guarda la posición de la carta que puede tirar, así como la puntuación. Esta puntuación es un valor igual a la resta de esa carta con el número 4, suponiendo que el 1 es el 0 y el rey el 9. Lo que realmente haremos es conseguir que cuanto más alejada esté una carta del 5, más ganas se tenga de tirarla.
Además, se utiliza otro bucle, que suma más valores a esta puntuación. El código es el que se muestra a continuación. Usa la función suma_puntuacion, que después se comentará.

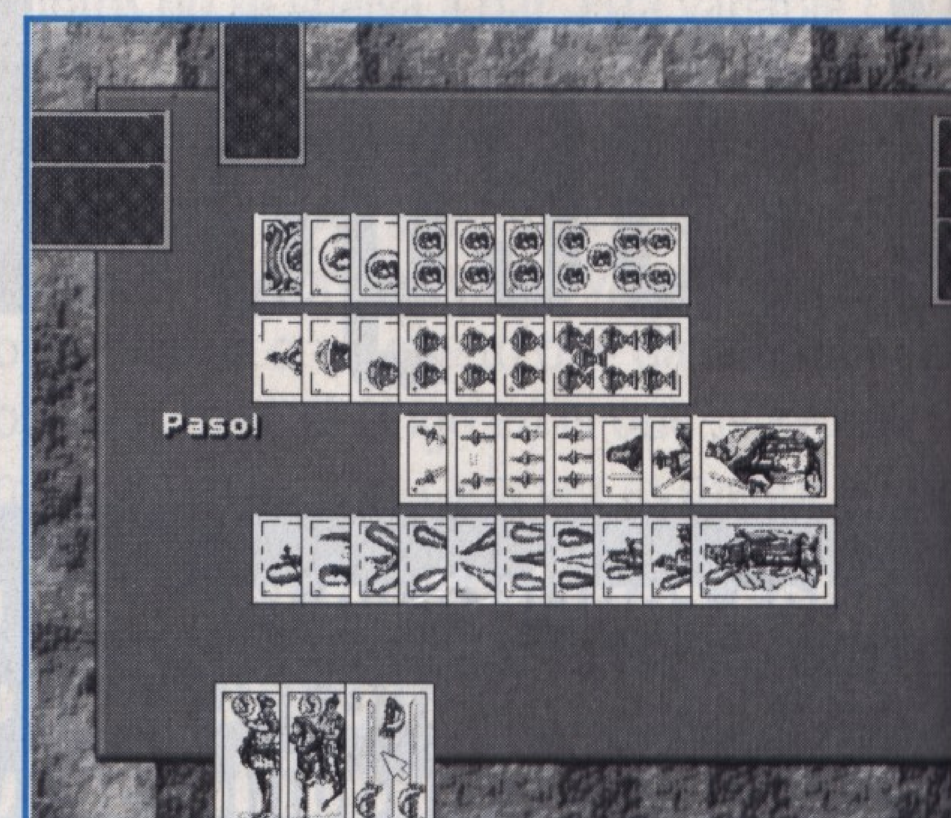


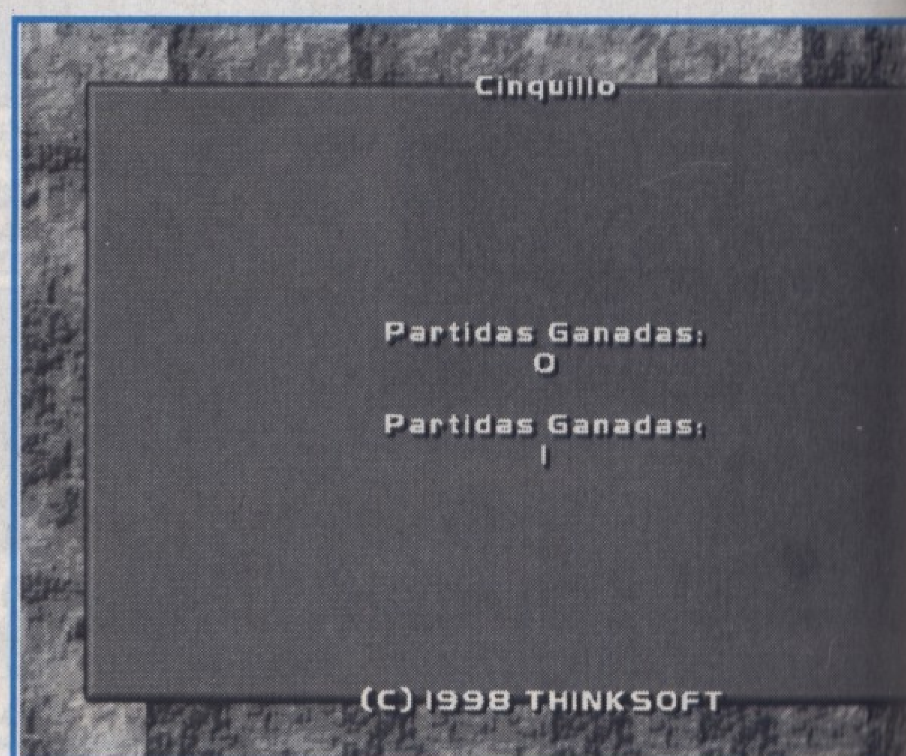
FIGURA 3.

```
FOR (contador1=0;contador1<ntiradas_cpu;conta
1++)
    puntosturno=suma_puntuacion(turno,
    jugador[turno].cartas[tiradascpu[contador1].
cionmano]);
    tiradascpu[contador1].puntuacion+=puntostu
END
```

Para realizar un programa ta sencillo lo más importante e saber las operaciones básica

La función suma_puntuacion() se encarga d sumar a la puntuación anterior todas las ca que tenga detrás. Es decir, si se tienen siete caballo y rey, a la carta siete le sumará el va hallado con el mismo método que el caso anterior, de las cartas del caballo y el rey. A se echará siempre la carta que más conven Después de este proceso sólo quedará orde las puntuaciones y elegir la más alta.
Los demás procesos no tienen ninguna complicación. Son sólo los comentados los se ocupan del juego de cartas en sí. Las tare que no se han comentado tratan del manej interactivo con el jugador, así como los me iniciales. Por último, comentar que la versió de este juego es freeware y no se prevén siguientes versiones.

FIGURA 4.



Signos de comparación para sentencias IF

==	Igual
<>(!=)	Distinto
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual

Utilizando la función Key() junto con la sentencia IF, podemos dotar a nuestros programas de control mediante el teclado.

En nuestro programa, el coche desaparece por el borde derecho de la pantalla. Si queremos que cuando desaparezca vuelva a aparecer por el lado izquierdo, sólo tenemos que añadir una sentencia IF. La condición sería: «si x es mayor que 640, entonces la nueva posición de x será 0». Expresado en código, sustituiríamos la sentencia process del coche por la siguiente:

```
PROCESS coche()
BEGIN
graph = 1;
x =100;
y=100;
LOOP
    IF (x>640)
x=0;
END
    FRAME;
END
END
```

Hay que notar que la posición expresada por las variables x,y se refiere al centro de la imagen. Las coordenadas del centro del coche, las podemos conocer pulsando la opción Info de la ventana del fichero *ejemplo1.fpg* y seleccionando la imagen del coche.

Otro tipo de condición es la siguiente:

```
IF (<variable>)
<sentencias>
...
END
```

Si la variable tiene valor impar, se considera la condición como cierta y se ejecutan las sentencias asociadas. Si es par, la condición no se cumple y, por tanto, no se ejecuta sentencia alguna. Esto es bastante útil para registrar la pulsación de teclas. Para ello se utiliza la función *Key(<código de tecla>)*. Algunas funciones devuelven valores, es decir, cuando se ejecutan, según el resultado de la operación, asignan el valor que devuelven a la variable indicada. La función *key()* devuelve 1 si la tecla está pulsada y 0 si no lo está. Lo vemos con un ejemplo:

```
Devuelve = Key(_esc);
```

Si la tecla ESCAPE está pulsada, *Devuelve* adquiere el valor 1, si no lo está , su valor será 0. Es decir, es como si se sustituyera en el código la sentencia *Key(_esc)* por el valor devuelto, resultando así: *Devuelve=1 (ó 0)*.

Los posibles códigos de teclado principales los vemos en el cuadro 5. Para más información véase el apéndice C.6 del manual de usuario de DIV.

Principales códigos de teclado disponibles en DIV

_esc	Escape
_f?:	Tecla de función F? (?=1..12)
_tab:	Tabulador
_enter:	Enter
_l_shift	Mayúsculas izquierdo
_r_shift	Mayúsculas derecho
_control	Teclas control
_alt	Alt o Alt Gr
_space	Barra espaciadora
_0..._9:	Números del 0 al 9
_a..._z:	Letras de a...z
_up:	Cursor arriba
_down:	Cursor abajo
_left:	Cursor izquierda
_right:	Cursor derecha
_backspace:	Borrado <=
_ins:	Insertar
_del:	Suprimir

Teniendo en cuenta lo anteriormente descrito, podemos conocer si una tecla, en este caso el cursor derecho, está pulsada mediante una sentencia *IF* de la forma:

```
IF (key(_right)==1)
x=x+1;
END
```

Si está pulsada, incrementa el valor de x. También podemos abreviar la expresión, atendiendo al valor devuelto por *key()*. Si está pulsada devuelve 1, es decir, un valor impar, y si devuelve 0, un valor par. Por tanto, aplicándolo al último caso estudiado de la sentencia IF, podemos abreviar de la siguiente forma:

```
IF(key(_right))
x=x+1;
END
```

Por tanto, ya podemos dotar a nuestro programa de control total con las teclas de cursor:

```
PROGRAM ejemplo1;

BEGIN
set_mode(m640x480);
load_fpg("ejemplo1.fpg");
coche( );
LOOP
    FRAME;
END
END
PROCESS coche()
BEGIN
graph = 1;
x =100;
y=100;
LOOP
IF (key(_right))
x=x+1;
END
IF(key(_left))
x=x-1:
END
FRAME;
END
END
```

Resultado del cambio de imagen por la pulsación del cursor. Con esto se obtiene mayor realismo en el movimiento, mirando hacia donde se mueve el coche.

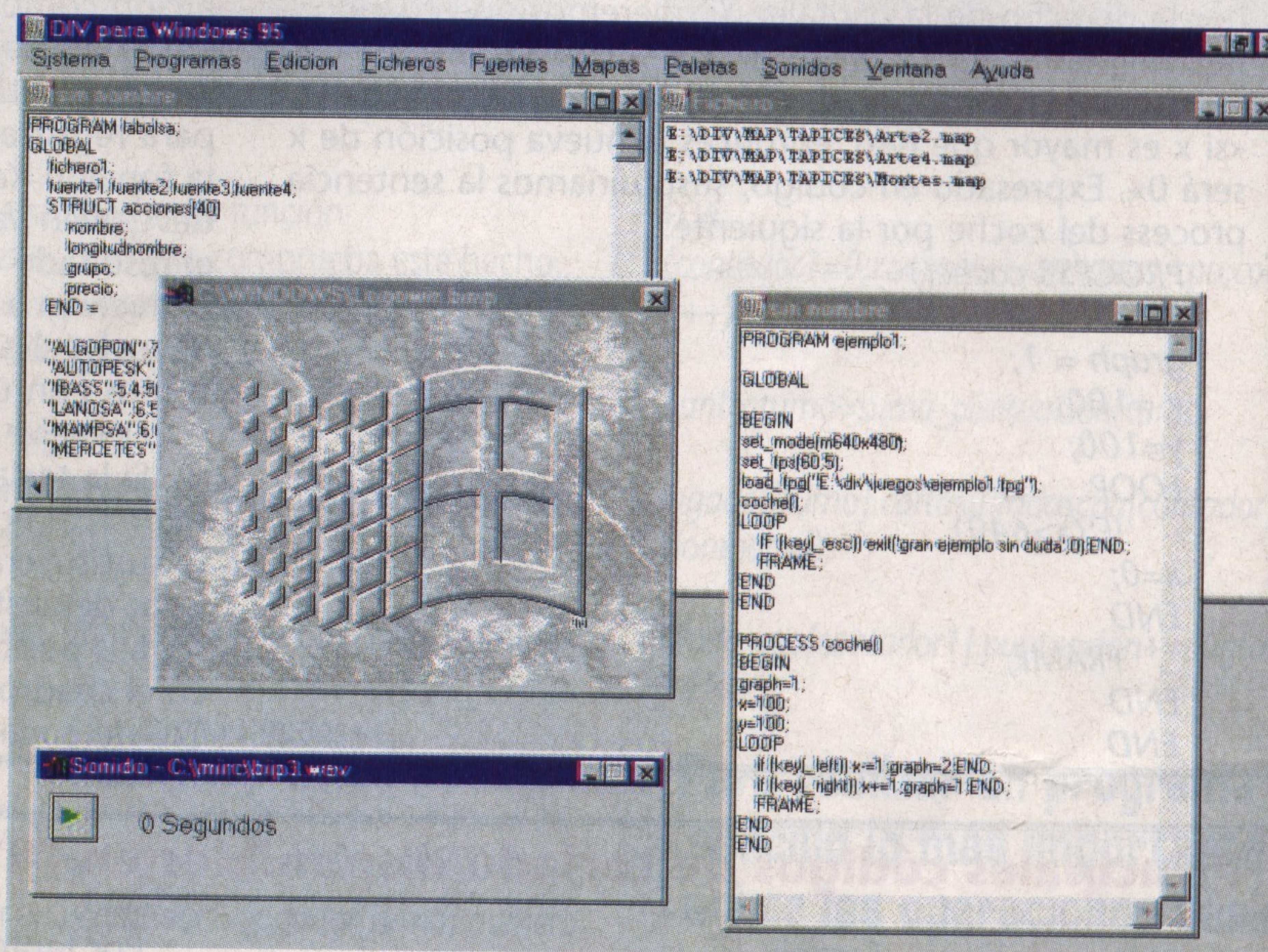
Es evidente que también podemos realizar otras muchas acciones con lo aprendido hasta ahora. Por ejemplo, si repetimos el proceso para crear el gráfico del coche, y creamos otro coche mirando hacia el lado contrario, y le asignamos el código 002, podemos cambiar la imagen del coche según la tecla pulsada con la sentencia:

```
Graph=2;
```

Dlls y DIV Games Studio p

DIV se prepara para el Sistema Operativo de Gates

Las Dll's son, tal vez, uno de los puntos fuertes del entorno de DIV. En esta sección aprenderemos a explotar todas las posibilidades del DIV mediante estos módulos. Además, hablaremos también acerca de un nuevo proyecto: DIV Games Studio para Windows 95.



Aspecto general que presenta la versión de Windows 95. Podemos ver sus características principales, como son los menús desplegables y ventanas completamente configurables.

DIV nació como una herramienta que pretendía hacer fácil la creación de videojuegos, aun sin tener conocimientos de programación. Pero tras esa cara inocente se esconde una de las herramientas más potentes de creación de juegos. En esta sección aprenderemos todos los entresijos del DIV, su estructura interna, así como la posibilidad de ampliar el lenguaje a través de bibliotecas de enlace dinámico, también denominadas Dlls.

Además, en este número hablaremos sobre la versión de DIV de Windows 95/98, en la que el autor de este artículo está trabajando. Cómo será y qué dificultades presenta el *port* a este sistema operativo, así como las afinidades y diferencias entre ambas versiones.

DIV y su estructura interna

Antes de afrontar la programación de Dlls, creemos conveniente conocer cuál es la estructura interna de los programas generados por DIV.

Los programas de DIV tienen un punto fuerte, y es la utilización de una librería externa o Dll, donde se concentran todas y cada una de las funciones del lenguaje DIV. Esta librería es *div32run.dll.* Con esto se evita la inclusión de todas aquellas funciones innecesarias en el código del ejecutable de nuestro juego, cargando en memoria sólo aquellas que son estrictamente necesarias. Con ello, el archivo ejecutable se convierte en un simple cúmulo de datos del juego y llamada a las funciones presentes en la librería *div32run.dll.*

Esto no parece ser relevante, pero esta posibilidad de llamar a funciones externas permite al lenguaje DIV no sólo acudir a la Dll principal, sino a otras Dlls de construcción propia o no. Con esto podemos ampliar el lenguaje DIV y dotar de nuevas capacidades a éste.

Aprenderemos cómo podemos utilizar las Dlls, las herramientas necesarias para construirlas, así como los distintos tipos. Con las Dlls podemos ampliar el lenguaje DIV y dotarlo de nuevas capacidades.

Tipos de DLLs

Distinguimos tres tipos de Dll's:

- Autocarga: son aquellas que pueden sustituir algunas de las funciones internas de DIV. Se denominan así porque se acoplan inmediatamente a cualquier juego sin necesidad de cambiar el código.
- Salvapantallas: al igual que los salvapantallas de Windows, son pequeños programas que transcurrido un tiempo de espera

Cuadro 1

Para construir Dlls, siempre que utilicemos Watcom C++, debemos añadir las siguientes líneas al archivo WLSYSTEM.LNK del directorio \BINB del compilador:

```
System begin div_dll
option osname="DIV DLL"
libpath %WATCOM%\lib386
libpath %WATCOM%\lib386\dos
format windows nt dll ^
end
```

De esta forma, tendremos disponible como formato de código el empleado por DIV(dll de Windows NT).

dio para Windows 95

muestran un efecto en la pantalla para volver a la situación anterior una vez pulsada una tecla, movido el ratón o joystick.
- De Funciones: añade nuevas funciones al lenguaje DIV.

Cómo utilizar DLLs

La utilización de Dll's no es complicada. Si la librería a utilizar es una Dll de autocarga o un salvapantallas, basta con añadir el fichero en cuestión al directorio donde se encuentra el ejecutable que va a utilizar dicha Dll. Si queremos que también funcionen en todos los juegos que ejecutemos en el entorno de DIV, tan sólo tendremos que añadir la Dll al directorio donde tenemos instalado DIV, es decir, donde se encuentra el ejecutable principal *d.exe*.

Para utilizar las Dlls de funciones, debemos cargarlas manualmente. Para ello, sólo tenemos que añadir la siguiente sentencia tras las declaraciones LOCAL del programa principal:

IMPORT "directorio\nombre.dll"

El directorio, por defecto, es aquel donde se encuentra el ejecutable del juego. Una vez hecho esto, todas las funciones que incorpora la librería estarán disponibles para el juego.

Construcción de DLLS

Así como la utilización de estos módulos es bien sencilla, su creación no lo es tanto. Para ello necesita de un profundo conocimiento del lenguaje C, así como de *buffers* y manejo de datos.

Las Dll's que utiliza DIV Games Studio son bibliotecas de enlace dinámico de Windows NT, por lo que es necesario tener un compilador de C que genere dicho tipo de archivo. El utilizado por el equipo de desarrollo de DIV es el Watcom C++ Compiler, aunque también puede compilarse con Borland C++ 4.X o superior. Para más información acerca de la utilización de este último compilador, visite la sección *downloads* de la página web oficial de DIV Games Studio (*www.divgames.com*).

El porqué del uso de estas Dlls es sencillo. Hacen uso de la tecnología de 32 bits, al igual que el entorno de DIV, aprovechando al máximo las capacidades del compilador. Además es un formato de alguna forma accesible a usuarios de nivel avanzado, que son los que se presuponen van a fabricar este tipo de módulos.

En este número vamos a aprender los principios básicos del desarrollo de Dlls de funciones. DIV Games Studio utiliza Dll's de Windows NT.

DLLs de funciones

En primer lugar, para poder utilizar todas las funciones de desarrollo de dlls debemos cargar el archivo de cabecera *div.h*, además de definir una constante de preprocesador llamada GLOBALS. Estas operaciones se realizarán después de cargar los archivos de cabecera necesarios para el módulo en cuestión. Lo vemos con un ejemplo:

```
#include <stdio.h>
#include <conio.h>

#define GLOBALS
#include "div.h"
```

Una vez realizado esto, estamos en disposición de utilizar todas las opciones que nos ofrece DIV. En primer lugar, debemos implementar dos funciones imprescindibles en cualquier módulo de este tipo. Su estructura es la siguiente:

```
void __export
divlibrary(LIBRARY_PARAMS) {
  //Aquí irán las declaraciones de
las funciones a exportar.
  }
  void __export divmain(COM-
MON_PARAMS) {
    GLOBAL_IMPORT();
  }
```

Tras esto, ya estamos en disposición de crear nuestra primera función para DIV. Vamos a elaborar una función que calcule el seno de un ángulo, bastante útil a la hora de realizar algunos efectos en demos. Hay que tener en cuenta que el valor que recibiremos indicará el ángulo en coma fija con tres decimales, es decir, 45,345 grados serán 45345 grados.

Crearemos una función llamada *seno()*. Recibirá un parámetro que indicará los grados del ángulo en cuestión. A la hora de crear Dlls, los parámetros no se escriben entre paréntesis sino que deben asignarse a variables en la propia implementación con la función *getparm()*. Asimismo la función debe declara como *void* el tipo de dato a devolver. Lo vemos mejor con el código de la función:

```
void seno()
{
int angulo=get-
parm();
}
```

Para utilizar correctamente las Dlls, únicamente hay que seguir con cuidado los pasos que señalamos

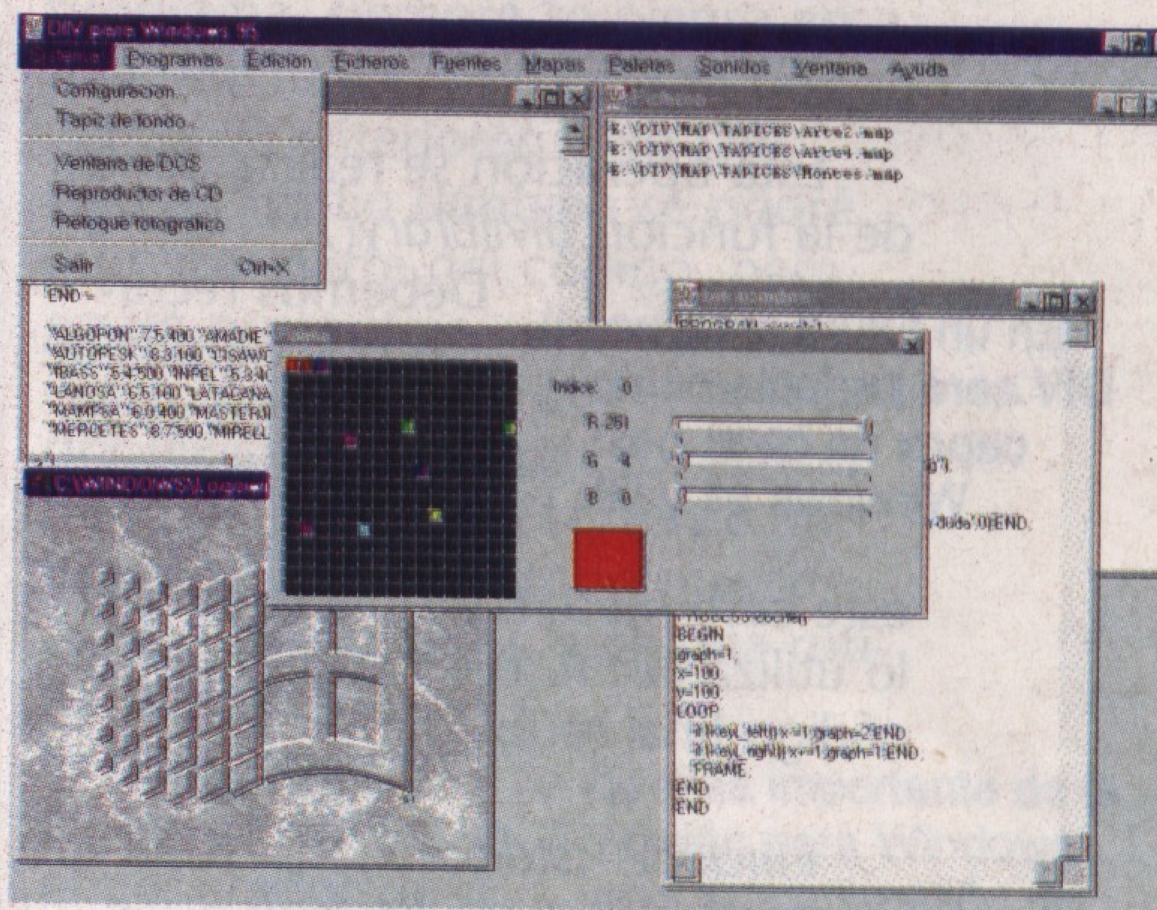
Todas las funciones deben devolver un valor aún si no es necesario. Para ello, utilizaremos la función *retval(int)*, para indicar el valor a devolver. En nuestro ejemplo, debemos convertir el ángulo dividiendo el parámetro recibido entre 1000 para, posteriormente, realizar la operación inversa con el resultado, tomando los tres dígitos más significativos. Para simplificar, tan sólo tomaremos los grados, sin decimales, por lo que no usaremos *floats*. La función completa sería:

```
void seno()
{
```

Cuadro 2

El archivo de cabecera *div.h* tiene un error en la declaración de la función *put_sprite()*. Para remediarlo hemos de sustituir dicha declaración por la siguiente:

void put_sprite (unsigned char si,int x, int y, //Put one sprite int an, int al, int xg, int yg, int ang, int size, int flags);*



Dentro del nuevo entorno, tendremos la posibilidad de configurarlo todo a nuestro gusto y añadir diversos links a cualquier programa que nos resulte de interés.

```
int angulo=getparm();
int resultado=int(sin(angu-
lo/1000)*1000);
retval(resultado);
}
```

Una vez realizada la implementación de nuestra función, debemos exportarla a DIV. Para ello utilizaremos la función *COM_export()*, que recibe tres parámetros:

- el primero indica el nombre que la función va a tener en DIV
- el segundo es el nombre de la función asociada, en nuestro caso *seno()*
- el último indica el número de parámetros que recibe dicha función.

En el ejemplo sería:

```
COM_export("SENO",seno,1);
```

Esta operación se realiza dentro de la función *divlibrary()*.

En un futuro próximo, DIV será perfectamente capaz de corer bajo Windows 95 y 98

Debemos recordar que hemos de llamar a los archivos de cabecera necesarios, en este caso *math.h* y *stdio.h* (ya que lo utiliza *div.h*). Por tanto, nuestro código final será:

```
#include <stdio.h>
#include <math.h>

#define GLOBALS
#include "div.h"
void seno()
{
int angulo=getparm();
int resultado=int(sin(angu-
lo/1000)*1000);
retval(resultado);
}
void __export
divlibrary(LIBRARY_PARAMS) {
  COM_export("SENO",seno,1);
}
void __export divmain(COM-
MON_PARAMS) {
  GLOBAL_IMPORT();
}
```

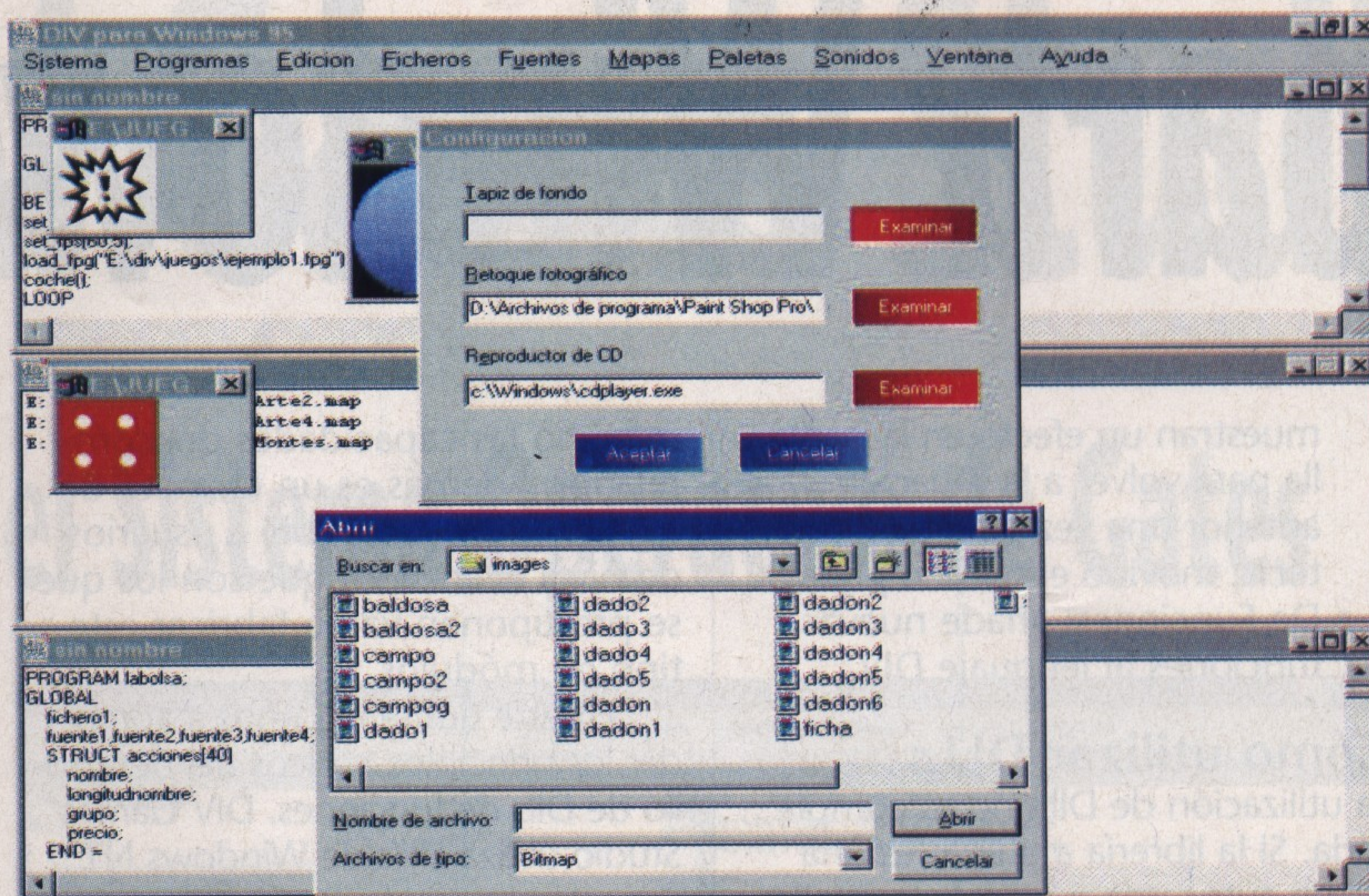
Guardamos el archivo de código con el nombre que deseemos, por ejemplo *seno.cpp*. Ya puede compilar la Dll y corregir los posibles fallos de compilación que le hayan surgido. Se dará cuenta de que hay un fallo en la declaración de la función *put_sprite* en *div.h*. Debemos sustituir la palabra *byte** por *unsigned char*. Con esto solucionaremos el problema y podremos continuar con la compilación.

Esto es todo por este número. Debemos practicar con funciones sencillas de este tipo y probarlas con programas de DIV. Asimismo, podemos utilizar como referencia el ejemplo *demo2.dll* incluido en el directorio dll de DIV. En nuestro próximo número profundizaremos más en la creación de Dll's.

Cuadro 3

Para compilar DLLs se recomienda el uso del archivo de procesos por lotes *make.bat*. Para ello, lo copiamos junto con *div.h* al directorio donde vamos a crear la Dll. Para compilar se escribirá lo siguiente:

MAKE <dll a compilar sin extensión>
Ej: Make hboy



El sistema de ventanas de Windows encaja perfectamente con DIV.

Los ejemplos del DIV

DIV Games Studio incluye ejemplos de la construcción de Dlls que pueden servirnos para entender la utilidad de estos módulos. Estos son:

- *ss1.dll*: salvapantallas que simula un fundido en forma de granos de arena
- *agua.dll*: realiza un efecto de agua en la zona inferior de la pantalla. Basta con incluirlo en el directorio de nuestro juego.
- *hboy.dll*: simula a la famosa Game Boy, ejecutando el juego en cuestión como en dicha consola. Un ejemplo cuyo funcionamiento merece la pena combrobar. También es una Dll de autocarga.
- *demo2.dll*: añade la capacidad de calcular raíces cuadradas a cualquier programa.

Así también podemos encontrar varios ejemplos de Dlls creadas por usuarios de DIV y cedidas como *freeware* en la página web oficial de DIV Games Studio, como por ejemplo *polys.dll* (dibuja líneas, cuadrados y círculos), *ascii.dll* (manejo de datos ASCII y entrada de datos por teclado) y *tad.dll* (manipula pilas y colas).

Hemos realizado una pequeña introducción al sistema de Dll's de DIV. En el próximo número continuaremos con la construcción de estos módulos tan importantes en nuestro entorno.

DIV mira al futuro

Por muchos es conocida la futura versión profesional de DIV en la que el equipo de programación dirigido por Daniel Moreno, programador principal de DIV, ha estado trabajando durante meses. Esta versión pretende cubrir todos aquellos huecos que no se trataron en la primera versión de este fabuloso entorno de desarrollo: 3D, redes y algunas otras sorpresas.

Pero muchos se habrán preguntado si algún día DIV trabajará en Windows 95/98. En estos momentos se está desarrollando una nueva versión del entorno de desarrollo de DIV para Windows 95. Este proyecto, supervisado por Antonio Marchal, miembro del equipo de desarrollo de DIV y llevado a cabo por Pablo Trinidad, tiene muchas expectativas de futuro. Aunque el entorno está casi acabado, aún existen muchos puntos importantes que desarrollar, como el compilador y otras herramientas asociadas al entorno, como el trazador de programas. No obstante, hemos puesto muchas esperanzas en esta nueva versión, que esperamos vea la luz lo antes posible. En este artículo informaremos de los últimos avances realizados en este proyecto, así como las nuevas características que incorporará y las dificultades que presenta el desarrollo de esta nueva versión de DIV.

DIV para Windows y sus novedades

En principio, esta nueva versión no va a incorporar grandes novedades

en lo que se refiere a utilidades del sistema, pero su código interno y estructura global va a cambiar de forma total.

El sistema de menús flotantes del entorno de DIV de la versión DOS se ve sustituido, en este caso, por el clásico menú de Windows, donde podremos acceder a las mismas opciones ofrecidas en DOS, pero de forma más rápida. Ya no es necesario tener en pantalla todas las ventanas que vayamos a utilizar abiertas. Con un simple clic en la barra de menús, todas las opciones estarán a nuestro alcance.

Además, se incluye un nuevo tipo de ventanas de herramientas, que pueden ser acopladas a los laterales de la pantalla, de forma que las opciones principales estén más a mano.

Otra de sus características fundamentales es su capacidad multitarea, que tan buenos resultados ha dado en otros programas de desarrollo. Esto nos permitirá trabajar con varios programas a la vez sin abandonar nunca el entorno de desarrollo. Podremos diseñar imágenes 3D con cualquier programa, editar sonidos y volver al entorno sin abandonarlo. Y por si no fuera poco, existen *links* directos a los programas complementarios utilizados por el usuario, completamente configurables.

Pero tal vez esto no sea demasiado novedoso. Con más o menos tiempo, se podía realizar lo mismo saliendo del entorno y volviendo a entrar, ya que la versión DOS se cargaba perfectamente desde Windows 95 sin salir de este sistema operativo, o acceder a las mismas opciones de menú, aunque fuera en forma de ventanas. Sin duda, lo más importante de esta versión será su generación de código.

En la actualidad, la casi totalidad de los juegos se desarrollan para Windows 95/98, y son pocas las compañías que hacen juegos para DOS. Las novedosas librerías para creación de juegos DirectX de Microsoft han supuesto una auténtica revolución en el mundo del videojuego. Ya no hay excusas para no programar en este nuevo sistema operativo, los juegos pueden funcionar casi tan rápido como en el DOS. Y por si fuera poco, no hay que configurar nada, es decir, tan sólo es necesario tener instalados los *drivers* de los distintos dispositivos instalados en nuestro equipo. DirectX se encargará de aprovechar al máximo todas las aceleraciones y características de nuestras tarjetas. Además, periódicamente Microsoft está sacando nuevas versiones más rápidas que incluyen nuevas capacidades.

Pero hay un pequeño problema. No todo el mundo sabe programar para Windows 95, y mucho menos para las DirectX. La documentación acerca de este nuevo sistema en nuestro idioma es muy escasa, por no decir nula, y hemos de tener un amplio conocimiento de inglés técnico para poder acceder con facilidad a todos los documentos que encontramos en Internet.

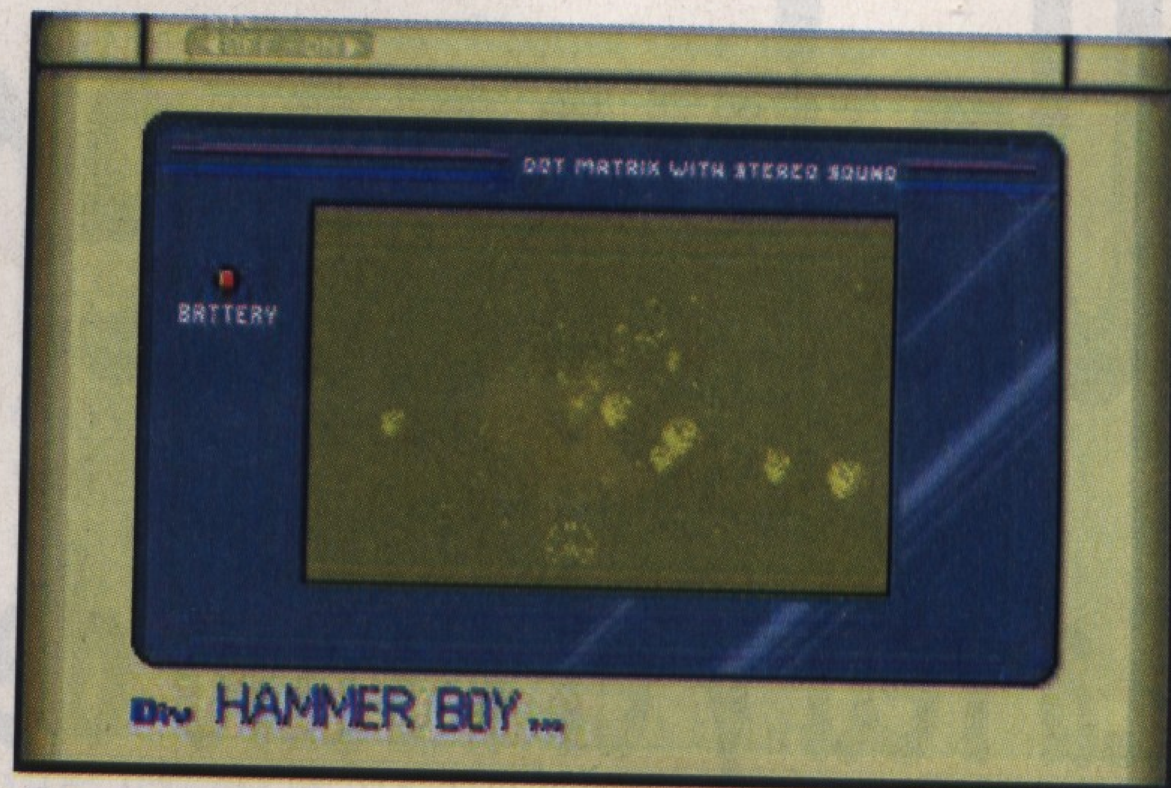
Esto, por supuesto, no ha pasado desapercibido al equipo de DIV, y han querido remediarlo también con esta nueva versión. Querían aprovechar todas las mejoras ofrecidas por este nuevo sistema. Quieren acercarlo a todos los programadores, conocedores o no de este sistema, para que aprovechar las últimas tecnologías esté al alcance de cualquier programador, y no sólo de los que se denominan gurús de la programación. Sin duda ,todo esto parece un tanto idealista, pero están poniendo todo de su parte para que este proyecto se haga realidad.

Ya que en este artículo hemos hecho referencia a las Dll's, no queremos olvidarnos de las nuevas capacidades de éstas. Ya que el código utilizado por los ejecutables de Windows 95 recurren a la API Win32, podrán implementar bibliotecas de enlace dinámico de Win95, con lo que cualquier programador de Windows 95 podrá crear estos módulos con cualquier compilador de este sistema operativo, mucho más frecuentes, incluso en otros lenguajes como pueden ser Delphi, Visual C++ y otros tantos. Además, nos permitirá actualizar el entorno DIV mediante nuevas bibliotecas de enlace dinámico, que se enlazarán al programa sin problemas.

El desarrollo de juegos con DIV en Windows 95 será mucho más cómodo.

Dificultades en el desarrollo

Lo anteriormente narrado suena bien, incluso parece convincente.



Con esta Dlls tus juegos te devolverán a los tiempos de Hammer Boy.

Pero detrás de todo esto hay una gran cantidad de inconvenientes y dificultades que surgirán a la hora de desarrollarlo.

Cuando se plantea realizar un *port* a Windows 95 de cualquier programa, siempre se piensa en una reutilización de código de versiones anteriores. Con DIV Games Studio también se podrá reutilizar parte del código, pero en un porcentaje mínimo. Es decir, tan sólo será útil el analizador léxico del compilador (encargado de traducir el texto de un programa para posteriormente crear el ejecutable), y alguna rutina de tratamiento de imágenes como puede ser el generador de explosiones. El resto del entorno deberá ser completamente nuevo, debido al uso de tecnologías tan distintas. Hay que rediseñar la Dll principal, *div32run.dll*, y cambiar todas las funciones incluidas en ella para traducirlas a Windows 95 y DirectX. Además, hay que contar con la aparición de la versión profesional de DIV, lo que aumentará el número de opciones y herramientas a incluir en la versión de Windows 95.

Lo más importante de la versión para Windows será su generación de código

La traducción del entorno DIV a un sistema operativo tan distinto y con tantas opciones hace plantearse añadir al lenguaje DIV opciones para el manejo más adecuado de Windows 95. No obstante, sólo se ha pensado realizar una ampliación, ya que la sólida estructura de DIV es independiente del sistema operativo que utilizar.

Conclusión

Esto ha sido una pequeña *preview* de lo que DIV ofrecerá. Debemos recordar que, sin las colaboraciones de los usuarios de DIV, no habría llegado tan alto. Con estos artículos mejorar el lenguaje DIV esté al alcance de más gente.

Pablo Trinidad

Un juego en tres dimensiones

La red y el sueño de las 3D, una realidad

En el presente artículo se presenta el primer capítulo de una serie que se va a dedicar al mundo de la programación de videojuegos en 3D. Por ahora, para empezar, se va a detallar qué necesitamos para programar en 3D y cuáles son las técnicas de programación gráficas 3D más usadas y conocidas.



El excelente Sin es una de las últimas aportaciones del género.

Este artículo es el primero de una serie que se van a presentar en esta publicación referidos al mundo de la programación de videojuegos en 3D.
Para introducir el tema, de momento se va a explicar qué conocimientos necesitan los que quieran meterse en este mundillo, cuáles son las herramientas y APIs más utilizados actualmente para programar videojuegos en 3D y, para acabar el artículo, se detallarán toda una serie de técnicas de programación gráfica 3D con las que es necesario que el lector vaya familiarizándose para saber qué es lo que deberá saber hacer y aplicar en sus propios programas en el futuro.

En lo referente a matemáticas, decir que es muy importante conocer algo de trigonometría y matrices.

Antes de empezar, sólo recordar que, para preguntas al autor, el lector puede mandar un E-mail a la dirección *erde@arrakis.es* o hacerlo en la página web: *www.arrakis.es/~erde.*



Qué necesito saber para programar en 3D

Aunque parezca lo contrario, para poder crear un programa que use gráficos 3D realmente no es necesario ser un experto de las matemáticas, aunque sí sería mejor que supieras algo de trigonometría, así como algo de matrices. Ello es así porque todos los cálculos 3D de los objetos se realizan con fórmulas que incluyen trigonometría, las cuales a su vez se obtienen en muchos casos de realizar unas simples matrices.
En lo referente a programación en sí, decir que para realizar un buen juego tridimensional, sabiendo bien el lenguaje C o C++ para Windows, cualquier persona puede ser capaz de meterse en la programación de un videojuego 3D.
Si por otro lado, es de los que usan el famoso paquete DIV para programar, hemos de decir que aunque el DIV actual está bastante limitado en lo que a programación 3D se refiere, hay que constatar que en el DIV Pro, el cual aparecerá al mercado en breve, sí que se pueden crear verdaderos videojuegos 3D.

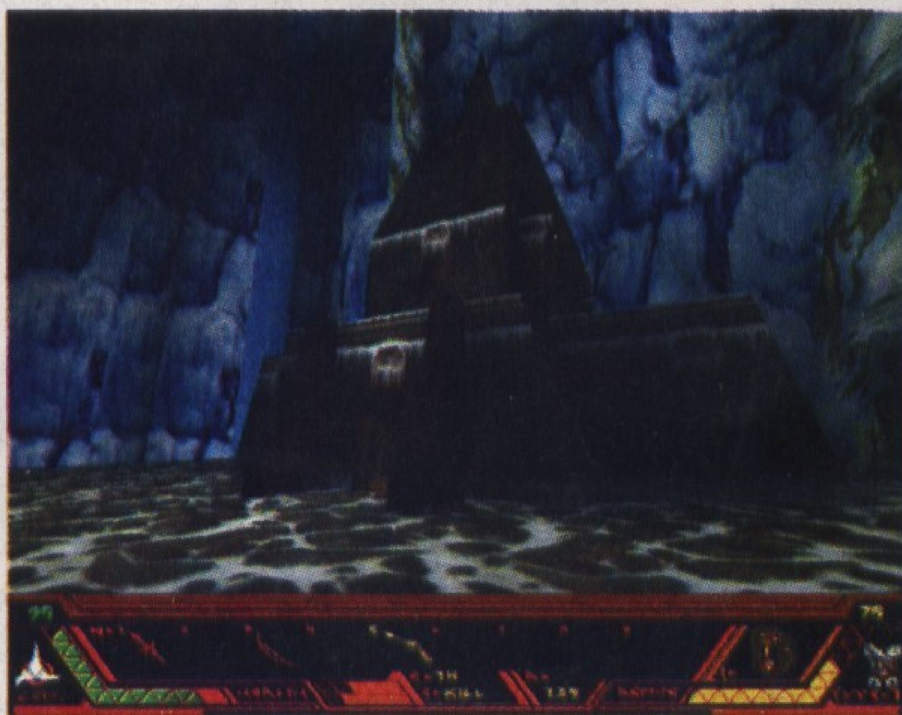
Sé lenguaje C/C++, ahora qué mas

Si el lector sabe programar en C/C++ para Windows, ahora viene lo nuevo: la posibilidad de crear videojuegos pasa obligatoriamente por aprender alguno de los API's de programación 3D estándar que existen en la actualidad.
El aprendizaje de un API 3D no es obligatorio realmente, pero se va a considerar en este artículo que ninguno de los lectores va a crearse el suyo propio, ya que ello sería una barbaridad técnica y que además no ofrecería ventajas, sino más bien lo contrario, desventajas.

Pero qué diablos es un Api 3D

Un Api 3D no es más que una librería de programación (*Aplication program Interface*) que contiene todas las funciones para la generación de gráficos 3D con toda clase de efectos ya creados. Nuestra función es *sólo* la de aprender a usar dicho Api y aprender a aplicarlo a nuestro propios programas.
Así por ejemplo, una vez dicho esto, tenemos que con cualquier API normal, una vez tenemos un objeto 3D calculado para dibujarlo, solamente tendremos que llamar a las funciones correspondientes que


Hasta la saga Star Trek ha tocado este adictivo género.


Con el tiempo, los mapeados son cada vez más complejos.

dibujarán los triángulos que lo forman , y el aplicarle efectos especiales de iluminación y sombreado será tan sencillo como invocar algunas otras pocas funciones de ese API.

Qué Api debería aprender

Ésta podría considerarse la pregunta del millón, ya que hay varios APIs estándar y a todos se les puede considerar buenos y de calidad. Pero para simplificarlo al máximo, vamos a resumirlo en que hay tres opciones posibles: el Direct 3D de Microsoft, El Open GL y el Glide (3Dfx).

• ***Direct 3D:***
El Direct 3D es quizás el API que se está imponiendo como un estándar de la plataforma PC y, por lo tanto, quizás es la mejor apuesta para nosotros. Es un API no demasiado fácil de programar pero que tiene la ventaja de que está en continua evolución (va por la versión 6.0) y que por lo tanto no se quedará nunca estancado, lo cual es también muy importante para en el futuro seguir evolucionando.

• ***Open GL:***
El Open GL es un estándar anterior al Direct X, y pese a que hay muy buenos juegos que permiten su uso, como el Quake II o el Unreal, es un API que no se sabe aún si se va a implantar como un estándar de PC. SI lo aprendemos, tendremos la ventaja de que es un API multiplataforma, por lo que si nos pasamos a programar en otro sistema operativo y plataforma, seguiremos pudiendo aprovechando nuestros conocimientos del mismo, puesto que seguro que también allí dispondremos del Open GL a nuestra disposición.

• ***Glide (3DFX):***
El Glide es el API de programación 3D más sencillo según los expertos de programar, pero tienen una seria limitación ya que el Glide está diseñado para programar las tarjetas aceleradoras tipo Vodoo y Vodoo2, por lo que necesitamos una tarjeta de este tipo instalada por fuerza para que nuestro programa funcione. Si el lector cree que va a diseñar su juego sólo para que funcione con aceleradoras y no por software, ésta es la mejor elección.

Técnicas más usadas en programación 3D

A continuación se detallan algunas de las principales técnicas gráficas usadas en la actualidad en la programación de gráficos 3D y que son necesarias conocer antes de meterse a programar un videojuego 3D:

• ***Efecto Fog:***
Este efecto (que se traduciría como *efecto niebla*) consiste en hacer que haya un efecto de niebla (atmósfera) y permite que, a medida que los objetos dibujados se alejan del observador, éstos se van difuminando más (nublando) hasta que llega un momento en que desaparecen de imagen. Este efecto hace que, si se aplica a todos los objetos de una imagen, dé la sensación de que existe realmente una atmósfera en el espacio que hay entre los objetos y el observador.
El efecto *fog* se puede regular en grado, que es como decir que se puede regular la densidad de la atmósfera y también puede seleccionarse el color del efecto *fog* (el color que da la supuesta atmósfera a los objetos).
Es una técnica muy usada en videojuegos donde hay escenarios exteriores y donde el campo de visión sin este efecto sería muy grande. Con esta técnica, como a partir de una distancia los objetos ya no son visibles, se hace que el programa se ahorre el tener que dibujar muchos decorados del fondo y así gana en velocidad.



• ***Luz especular:***
Otro efecto que se puede aplicar en los gráficos de un juego 3D es el de luz especular, lo que hace que los objetos brillen con uno u otro color dependiendo del color que se atribuya a la fuente de luz y del color del propio objeto sobre el que *rebota* la luz, haciendo por ejemplo que una pelota roja apareciera de color azulado bajo una iluminación de color azul.
Ésta es una técnica bastante compleja, aunque los juegos que la utilizan aumentan el realismo de los gráficos en gran medida.

• ***Alpha Blending:***
El atributo *alpha* nos da la posibilidad de que cada objeto pueda tener atribuido un grado de transparencia más o menos elevado (desde opaco total hasta transparencia total).
El poder aplicar transparencias nos permitirá realizar efectos especiales asombrosos, como son por ejemplo los paneles de información que aparecen en el juego Unreal, donde se ven hologramas de dispositivos futuristas.
También se puede usar para crear efectos de los disparos, para dibujar fantasmas, efectos especiales como humo, etc.

• ***Mezclado de color:***
Este algoritmo hace que cuando se unen varios valores en un mismo píxel físico de color, el chip une los colores y escribe el resultante de su mezcla. Esto es muy útil para mejorar la calidad de las imágenes. Imaginemos una escena 3D cualquiera con un pasillo muy largo. Si aplicamos esta técnica a objetos que están al fondo de todo de la escena, ello hará que los puntos se mezclen y no den la sensación de superposición, lo que hará que cuando nos acerquemos hacia el fondo, las cosas aumenten de tamaño con colores más suaves y no de forma irregular y errática.

• ***Sombreado de polígonos:***
Hay dos técnicas principales de sombreado de polígonos:, el tipo *flat* y el *gouraud.*



El primero, el *flat*, hace que cada polígono posea un único color de sombreado, lo que hace que se puedan diferenciar cada uno de los polígonos cuando hay cambios de brillo muy pronunciado entre polígonos adyacentes de una superficie. Es la técnica menos realista en cuanto a resultados, aunque también la que necesita menos tiempo de procesador para dibujar los triángulos,

El otro método, el *gouraud*, hace que el color de brillo de los polígonos quede difuminado, haciendo que los extremos de los triángulos queden unidos entre ellos por el mismo nivel de brillo, lo que da una sensación de mayor realismo y más calidad a la imagen. El uso de la técnica *gouraurd* hace que los juegos sean mucho más realistas ya que disimula los triángulos que forman los objetos y por lo tanto quedan más *reales*.

El filtrado de texturas es una de las técnica más importantes para generar decorados de calidad.

• ***Color Keying***:
La técnica del color *key* consiste en hacer posible que a una determinada textura se le atribuya un color como transparente y poder hacer así que zonas determinadas queden transparentes a la hora de aplicarse dichas texturas
Esta técnica es muy usada y puede ser práctica a la hora de hacer, por ejemplo, ventanas, puertas o similares donde haya una parte que tenga que dejar ver lo que hay detrás.
Es una técnica simple, como la que se usa en el dibujado de sprites 2D, donde hay 1 color que hace de transparente, pero que a su vez es muy útil para poder hacer decorados mucho más realistas.
Para ver ejemplos de esta técnica, podemos referirnos a programas tan veteranos como es el *Doom*, *Quake* o cualquier otro de la serie.

• ***Filtrado de texturas:***
El filtrado de texturas son unos algoritmos muy apreciados por los usuarios y consiste simplemente en mecanismos que hacen que las texturas aparezcan lo mejor dibujadas posibles a cualquier distancia (tamaño) en que aparezcan.
Hay varias técnicas de filtrado de texturas, aunque las más conocidas son quizás las de *nearest* y la lineal (conocido por el nombre de *bilineal*).
Las técnicas de filtrado de texturas son algo complejas, aunque la verdad es que un videojuego sin uso de ninguna técnica de éstas dejará mucho que desear en cuanto a calidad de imagen, sobretodo cuando haya escenarios con bastante campo de visión ya que a más pequeño se vea todo, peor quedará dibujado si no los usamos.

• ***Técnica Mipmapping:***
Este mecanismo es muy usado y conocido y se utiliza para que las texturas aparezcan dibujadas siempre con una buena calidad sea cual sea la distancia a la que se dibujen en pantalla. Básicamente este sistema consiste en que cada textura se guarda en memoria en varias versiones de tamaño (escalas) y cuando el programa aplica la textura a un objeto en imagen, usa como textura a aplicar automáticamente la que más se ajusta al tamaño con el que aparecerá en pantalla.
Es una técnica relativamente sencilla de aplicar, aunque multiplica el espacio necesario en memoria para almacenar los gráficos de texturas, pero con los ordenadores actuales que posean 32, 64 o 128 Megabits de memoria RAM esto no representa ningún problema.

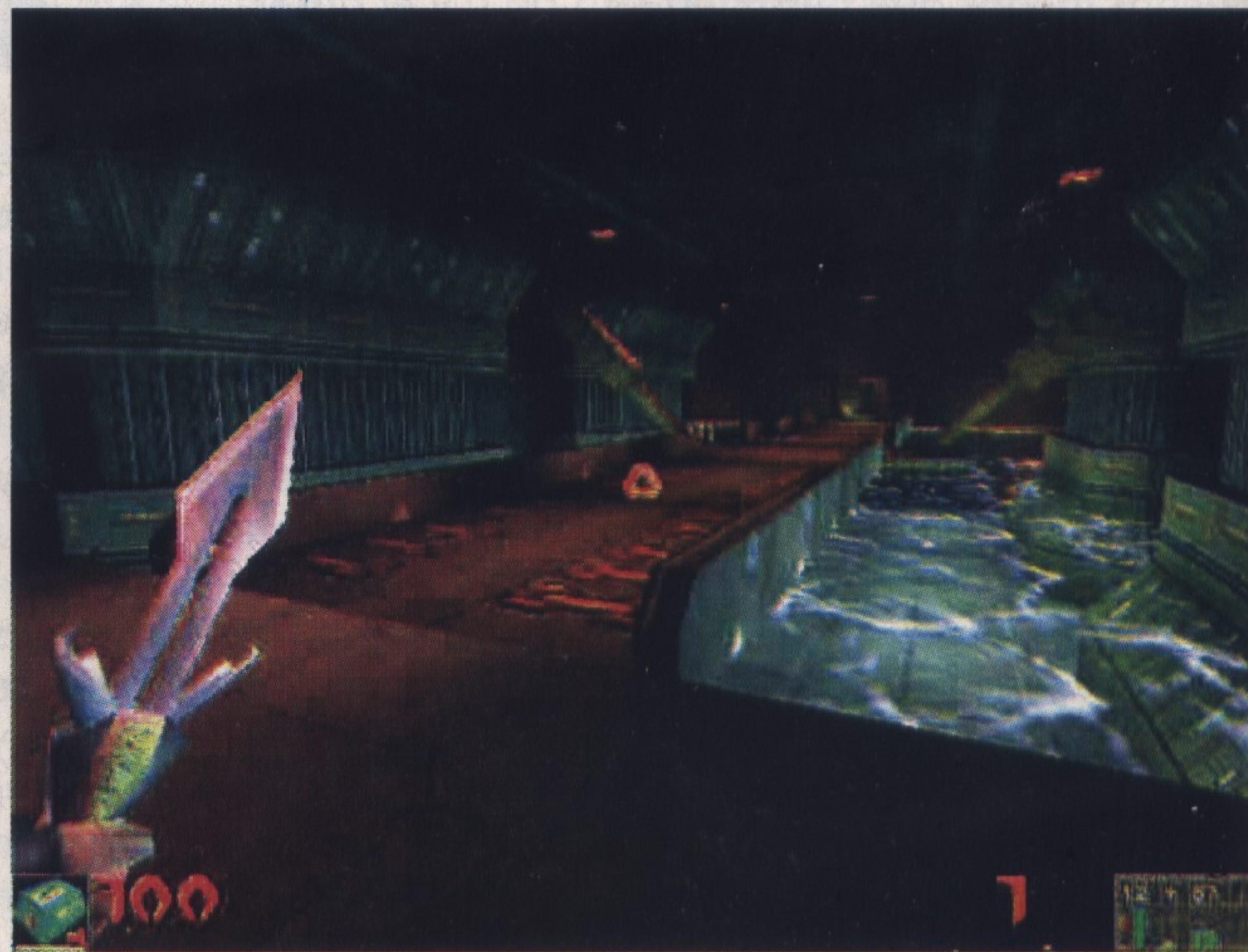
• ***Antialiasing:***
Este algoritmo permite hacer que los triángulos de los gráficos que se dibujan en pantalla aparezcan con los bordes exteriores sin que se note el típico efecto de pixelación (dentado), de forma que se suavizan los ángulos y la imagen aumenta en calidad y definición.
Esta técnica es muy útil cuando se programan juegos que han de funcionar en modos de poca resolución, como es por ejemplo a 320*200 o 640*480, ya que permite que las imágenes aparezcan con una apariencia mucho más definida.

Enrique de Alarcón





La posibilidad de descubrir por ti mismo la complejidad de los escenarios es uno de los alicientes de los arcades 3D.

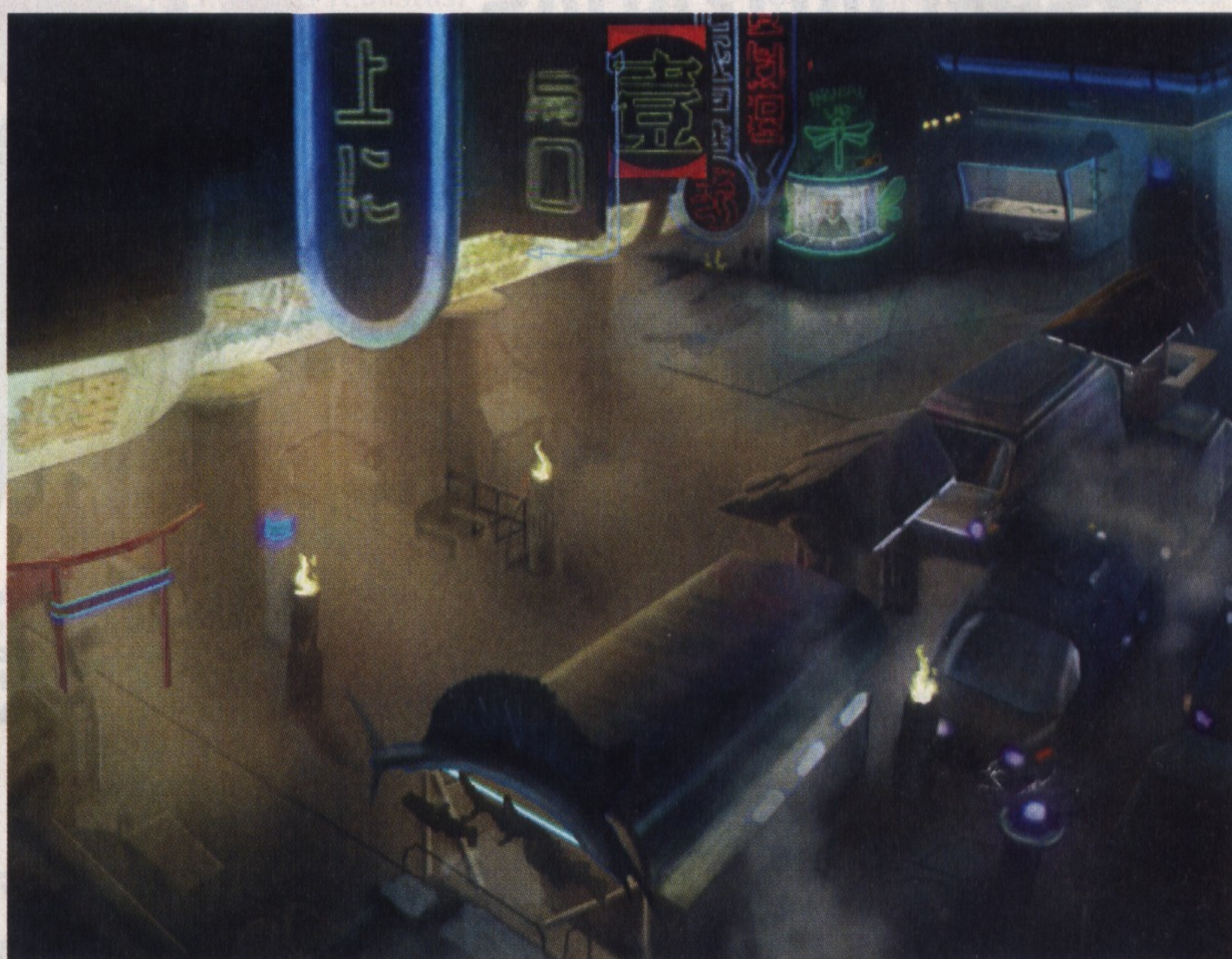


No te apures: siempre sabrás cómo te encuentras y de qué dispones si observas la parte inferior de la pantalla.

El color de la aventura

Cómo entrar en un mundo fascinante

Transporta a tus amigos a un mundo de fantasía de tu propia creación y proporciónales algún quebradero de cabeza. Salva a tu princesa favorita, mientras unos piratas se embriagan de grog. Y claro, todo ello con sólo usar el ratón.



En la genial videoaventura Blade Runner los escenarios se cuidaron al máximo. Por ello la calidad gráfica permitió que los ambientes se asemejaran enormemente a los de la película.

De los muchos tipos de juegos que existen para ordenador, uno de ellos son las aventuras. En este tipo de juegos, el participante se ve totalmente inmerso en un mundo de fantasía creado por el programador. Ejercitando su buen juicio, su inteligencia y su conocimiento de los hechos y personajes que encuentre, recorre el mundo intentando resolver el enigma al que el programador lo ha desafiado.

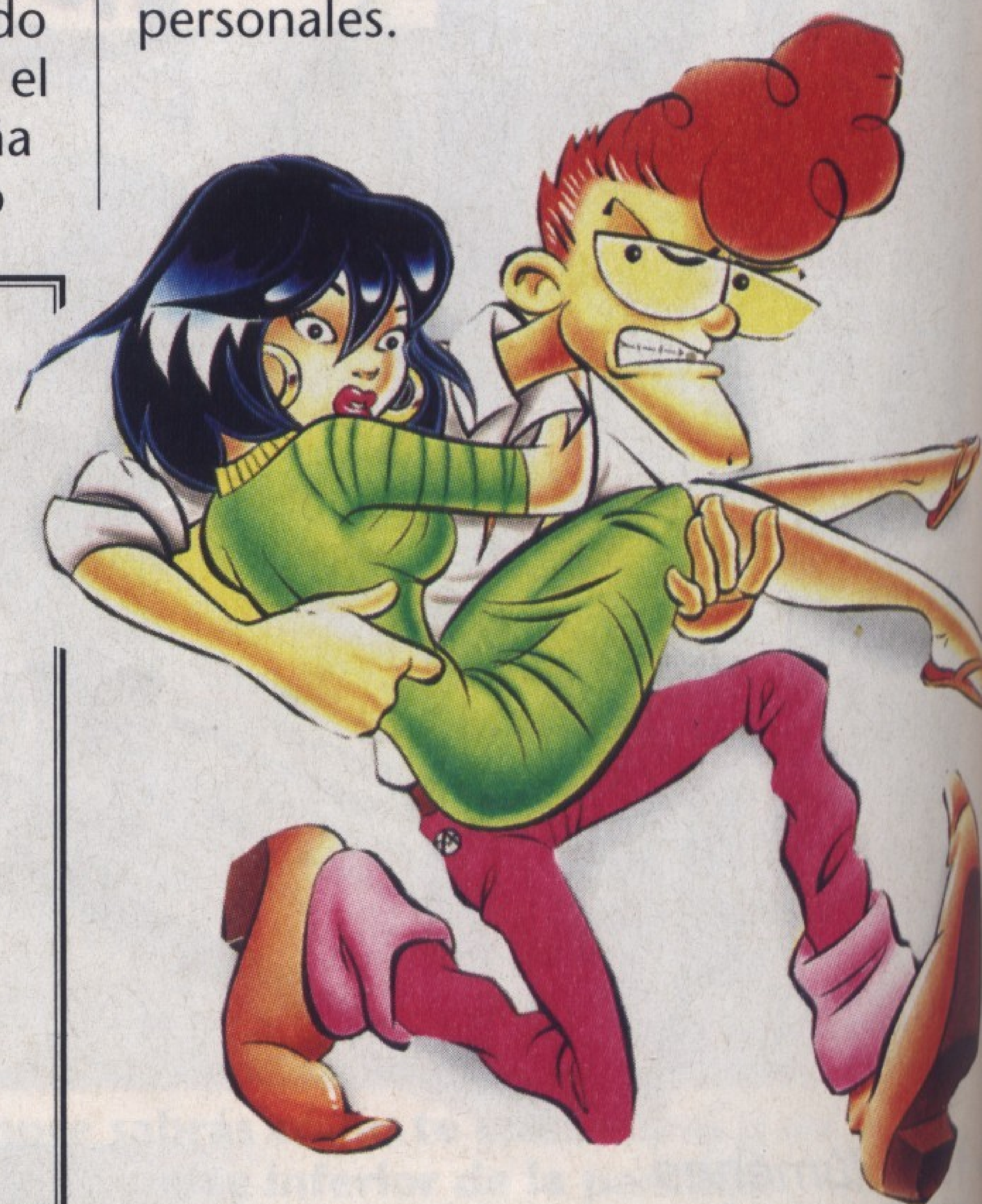
Para la creación de cualquier tipo de videoaventura hay que seguir unas pautas que ha marcado el propio género a lo largo de su historia. Encualquier caso, el germen es la idea, por muy simple que sea, que da lugar al desarrollo del argumento.

Historia

La idea de escribir juegos de aventuras procedía originalmente de la popularidad de juegos como *Dragones y Mazmorras* (*Dungeons & Dragons*, también llamado D&D), y el deseo de utilizar ordenadores para algo más que el mero proceso de datos. En *Dragones y Mazmorras*, el jugador adopta una determinada personalidad y penetra (con su imaginación) en un mundo conocido como la *Mazmorra*, creado por el *Carcelero*. En los juegos de aventuras el programador adopta un papel similar al Carcelero, creando un mundo propio, mientras que el jugador, por su parte, desempeña un papel semejante al de alguno de los personajes del juego. A diferencia del juego tradicional, los jugadores de aventuras por ordenador no pueden elegir normalmente los rasgos de su carácter. De esto se ocupó el programador a la hora de escribir la aventura. En algunas versiones sofisticadas, se le permite al jugador seleccionar su equipo o sus rasgos personales.

Pasos en el diseño de una aventura

- Elegir un tema que pueda ser interesante. Plantear un argumento.
- Recoger ideas, cuantas más mejor.
- Seleccionar y ordenar las ideas.
- Preparar la historia completa de la aventura paso a paso.
- Elaborar un mapa y dividirlo en «habitaciones». Enumerarlas. Sitúar los personajes y objetos.
- Determinar el orden en que deben ocurrir los sucesos y las condiciones que pueden darse.
- Definir una cadena de congruencias para que el juego sea lógico.
- Asegúrarse de que todos los pasos anteriores se han completado correctamente.
- Sentarse delante del ordenador y... a trabajar.



La primera aventura se escribió en un gran ordenador usándose FORTRAN. El programa ocupaba 300Kbytes de memoria. Sin embargo, el verdadero principio fue debido a Scott Adams, que trasladó algunas ideas al célebre TRS 80, en 1.978. Con esto logró demostrar que era posible escribir un juego de aventuras totalmente satisfactorio precisando menos espacio de memoria. Desde entonces, los temas de aventuras que Adams adaptó para sus juegos (Aventurilandia, La Cueva del Pirata, El Misterio de la Casa Encantada) han sido utilizados en multitud de ocasiones.

Las aventuras conversacionales aparecieron en España a principios de los ochenta, gracias a la gran aceptación del ordenador doméstico. Estos ordenadores solían contar con un microprocesador Z-80 y no superaban los 64Kbytes, siendo lo habitual 48Kbytes, en donde las aventuras solían estar programadas en BASIC. La afición a las aventuras fue creciendo en España, y cada vez había más jugadores y programadores. Empezaron a aparecer las historias gráficas ilustradas, primero con textos (dibujos ASCII) y después con imágenes. Hasta bien avanzada la era del ordenador personal (PC), con la evolución de las primeras tarjetas gráficas EGA, las aventuras gráficas no llegaron a estar al alcance de todos.

Tipos

Tanto el juego original para el gran ordenador como los juegos de Adams, para un microordenador, únicamente presentan texto sobre la pantalla. Este tipo de aventuras, en las que no aparecen gráficos, siguen siendo populares y hay quien afirma que son las mejores, ya que la imaginación del jugador no la iguala ningún gráfico. Más tarde, surgieron aventuras en las que las descripciones aparecían acompañadas de una imagen ilustrativa; pero las más populares son las aventuras gráficas. En éstas, el personaje que adopta el jugador es un «muñeco» que se desplaza por un escenario gráfico con el que interactúa constantemente. En la actualidad existen juegos para ordenador denominados de ROL, que se ajustan con más fidelidad a la idea del juego tradicional D&D. Estos juegos, debido a sus propias características, no son comparados ni mezclados con las aventuras gráficas. Otra evolución de las aventuras conversacionales de sólo texto han sido los MUD (*MultiUser Dungeon*), que consisten en juegos multiusuario para redes (sobre todo Internet). El primero fue realizado entre 1.979 y 1.980 para Internet por Richard Bartle y Roy Trubshaw, el MUD1, siguiendo también el patrón de *Dragones y Mazmorras*. Estos juegos se apartan demasiado de la idea de una aventura. No hay nada que resolver para ganar el juego, solamente hay que ir adquiriendo experiencia (puntos) para ir ascendiendo de nivel. A la vez que sumamos puntos podemos dialogar con otros jugadores, pedirles ayuda o ayudarles, incluso aliarse para tener más probabilidades de éxito.

Jugando aventuras

Cuando se inicia una aventura, normalmente, se hace una introducción que sitúa al jugador en el juego: lugar en el que se encuentra, qué ha ocurrido, qué debe hacer y para qué. Durante el juego va recorriendo salas o escenarios que le son descritos (en las de texto e ilustradas), o mostradas (en las ilustradas y gráficas). En cada una de ellas se pueden realizar varias acciones como mirar cosas, coger objetos, usarlos, etc... En las aventuras de texto e ilustradas esto se hacía escribiendo las ordenes directamente como: *coger bastón, usar bastón contra bicho feo o ir norte.* En las aventuras gráficas hay diversos métodos en lo que todo se reduce a clicks de ratón.

Resolver aventuras

Normalmente, sólo hay una solución para cada aventura, aunque cada vez son más las aventuras que tienen distintas formas de resolverse.

Los objetos que se van encontrando a lo largo del juego sólo suelen usarse una vez y siempre son útiles. Existe una regla de oro para todo jugador de aventuras: «*coge todo lo que puedas, prueba todas las combinaciones*». No siempre está claro la utilidad de los objetos, es posible que necesitemos una escopeta para alcanzar un libro de una estantería muy alta. A veces, los objetos son armas de doble filo. Quizás necesitemos una bolsa de oro para pagar el peaje si queremos pasar un puente, pero si por casualidad optamos por cruzar el río a nado, puede ser que el peso del oro nos hunda.

Hasta las más originales videoaventuras toman como base unos personajes tipo que siempre se repiten

Una práctica habitual es trazar un mapa a la vez que se juega. Anotar las distintas habitaciones, salidas, objetos y personajes, incluso peligros. En las aventuras, quizás porque una imagen vale más que mil palabras, los jugadores sólo recurren a un mapa escrito si están desesperados en un laberinto. Salvo excepciones, gracias a las imágenes los jugadores suelen retener bastante bien un mapa mental del juego.

Descripción de los objetos y habitaciones del mapa

- Ha1.Pasillo
  - Ob1. Cuadros. Tras uno está la llave.
  - Ob2. Llave. Para abrir la puerta.
  - Ob3. Máquina de refrescos. Para comprar una refrescante lata.
  - Ob4. Puerta de la habitación. Para entrar en ella.
  - Ob5. Salida. Para ir a la plaza.
- Ha2.Habitación
  - Ob6. Teléfono. Es una simple distracción.
  - Ob7. Compañero de habitación. Principio y fin del juego.
  - Ob8. Cartera. Tiene el dinero necesario para las compras.
  - 0b9. Monedas. Para usarla con la máquina de refrescos.
  - Ob10. Camas. Son adornos para la habitación.
  - Ob11. Mesilla. Igual que las camas y el teléfono.
  - Ob12. Puerta. Para salir al pasillo.
- Ha3.Plaza de la residencia
  - Ob13. Edificio principal. Para volver al pasillo.
  - Ob14. Jardinero. Nos dará pistas sobre el juego.
  - Ob15. Fuente. Un adecuado adorno.
  - Ob16. Camino. Nos lleva al cobertizo.
  - Ob17. Verja. Limita con la calle.
- Ha4.Cobertizo del Jardinero
  - Ob18. Pelota. Para los niños.
  - Ob19. Escalera. Para alcanzar la pelota.
  - Ob20. Camino. Para volver a la plaza.
- Ha5.Calle
  - Ob21. Niños aburridos. Tienen la foto y quieren la pelota.
  - Ob22. Verja. Limita con la plaza.

Crear nuestra propia aventura

Escribir una aventura es una buena manera de ejercitarse en la programación, ya que se tocan todos los aspectos importantes de cualquier lenguaje. Por otro lado, se trata de tipos de juegos muy populares y agradecidos.



Antes de sentarse a escribir una aventura hay que tener muy claro lo que se quiere hacer. Improvisar sobre la marcha puede acarrearnos muchos quebraderos de cabeza. Lo mejor es tomar papel y lápiz e ir plasmando ideas a las que luego daremos forma. Al principio son un montón de frases que aparentan no tener sentido, pero conforme lo vayamos creyendo necesario seleccionaremos las mejores, descartaremos las inadecuadas y añadiremos otras.

Lo primero que necesitamos es una historia que motive la aventura gráfica. Después, definiremos el mapa o Mundo y, por último, concretaremos las Congruencias.



La historia

Primero se define el argumento de la aventura: «Adoptamos el papel de un estudiante que hace su vida en el campus de la universidad. Un día nuestro compañero de habitación cae en una gran depresión porque necesita una foto de su musa, que ha perdido, para realizar un cuadro que presentar a su profesor de Artes Gráficas. Nuestra misión: salvar a nuestro compañero recuperando la foto». En base a este argumento se recogen ideas que, poco a poco, forman la historia completa: «Hemos olvidado la cartera en la habitación y no podemos entrar. Cogemos la llave que escondemos tras el cuadro del pasillo (para estas ocasiones) y entramos. Nuestro compañero está desmoralizado y nos introduce en el juego contándonos lo que debemos hacer. Cogemos nuestra cartera de la cama y salimos. Nos encontramos con unos niños que tienen la foto y la usan para divertirse. Sólo nos la darán si le devolvemos la pelota que han perdido en un tejado. Vamos a por la pelota usando la escalera para alcanzarla, pero descubrimos que está pegada. Para despegarla buscamos una máquina de refrescos y compramos una lata. La usamos con la pelota y esta cede. Se la cambiamos a los niños por la foto y devolvemos la foto a nuestro compañero

El mundo o mapa

Siguiendo la historia, dibujamos un mapa del mundo y lo dividimos en habitaciones. Las numeramos y situamos objetos y personajes.

Cadenas de congruencias

Para que el juego tenga coherencia, las acciones han de ser congruentes. Nadie puede abrir una puerta, aunque tenga la llave, si antes no ha usado la llave con la puerta. Las cadenas de congruencia ayudan a conseguir que el juego tenga sentido. Establezcamos un orden:

INICIO (pasillo) -> Mover cuadro -> Coger llave -> Usar llave con puerta de la habitación -> Abrir cama -> Coger cartera -> Mirar cartera -> Hablar con compañero de clase -> Hablar con niños de la calle -> Coger pelota -> Comprar refresco -> Abrir refresco -> Usar refresco con pelota -> Coger pelota -> Dar pelota a niños -> Dar foto a compañero de clase -> FIN

Determinamos las condiciones:

- No sales a la plaza sin la cartera
- Se podrá ver y coger la llave si se movió el cuadro
- Si usas la llave, abres la puerta
- Se podrá ver y coger la cartera si se abrió la cama
- Se tiene monedas si se miró la cartera cuando la teníamos (la cartera queda vacía)
- Sabemos que los niños tienen la foto si los miramos después de hablar con el compañero
- Sabremos que hay que coger la pelota si le decimos a los niños que queremos la foto
- Sólo usas la escalera será si sabes que debes coger la pelota
- No podremos comprar refrescos hasta probar a coger la pelota

Finalmente, realizamos la cadena de congruencias. Comienza en la izquierda y finaliza a la derecha. Marcamos con trazos rojos las sucesiones obligatorias y con amarillo las opcionales. Los fragmentos de las cadenas paralelas pueden resolverse a la vez. Si dos o más segmentos convergen, los sucesos de los segmentos que convergen han de haber ocurrido para continuar. Cuando un segmento diverge en varios origina segmentos paralelos.

Miguel Adolfo Barroso

Consejos para diseñar una aventura

- Informar. No hay que olvidar informar al jugador de todo lo necesario para iniciar el juego. Es totalmente cruel que se deba encontrar un tesoro para completar la aventura si nunca se le ha dicho al jugador que debe buscarlo.
- Mundo. Ser cuidadoso con el mundo a elegir. Hay que intentar que sea muy interesante si se quiere hacer una aventura apasionante. Es muy difícil conseguir que sea excitante buscar una goma de borrar en un edificio de oficinas.
- Equilibrio. Buscar el equilibrio adecuado entre desafío e imposibilidad. No es bueno gastar tiempo de dedicación en un juego que puede resolverse en dos horas, pero tampoco harás muchos amigos si no es posible completarlo.
- Mantener la tensión. Para hacer más grande el Mundo se suelen añadir habitaciones o escenarios muertos (no útiles o vacíos). No se debe abusar de ellos, agotan recursos y aburren a los jugadores.
- Incrementar el nivel. En tus primeras aventuras no se debe apuntar demasiado alto. Se debe comenzar por aventuras pequeñas y, a medida que se dominen las técnicas, hacer otras mucho más complejas.

Próximo número

En la próxima entrega daremos un paso más en la creación de videoaventuras... Muy pronto todos seremos capaces de hacer un título similar.



La de Alfred fue una aventura al más puro estilo de los clásicos, con multitud de posibilidades para los expertos.

Juegos y estrategia

Todos los secretos del género



Una de las últimas aportaciones en el campo de la estrategia.

En esta sección vamos a seguir un curso de programación de juegos de estrategia. Empezaremos recopilando las cosas que tienen en común los juegos de estrategia, o lo que es lo mismo, qué contribuye a que un juego sea considerado de estrategia.

Para aprender a hacer una cosa lo mejor es hacerla. Así que a lo largo de este curso vamos a realizar un juego de estrategia, eso sí, muy simple, ya que para complicarlo ya está cada uno. Iremos haciéndolo por pasos, es decir, lo diseñaremos, aprovechando para comentar lo que tiene todo juego de estrategia, iremos codificando las rutinas más fáciles y generales, como los menús, botones, ratón, entorno. Luego pasaremos a los procesos más complejos, tales como búsquedas de caminos, desde los más sencillos hasta los más complicados. Por fin, le daremos forma al juego, para que sea distinto de todos los demás.

Y ahora, empecemos con el diseño...

Diseñando un juego

Lo primero que debemos saber para programar un juego de estrategia son las características que debe cumplir un juego para ser calificado de estrategia. Sino, lo más probable es que nos salga algo parecido a un arcade. Lo primero de todo es el objetivo, ya que no siempre es la destrucción del enemigo. Puede ser rescatar una unidad capturada, conquistar una determinada edificación, o infiltrar algún espía en la base enemiga. Para conseguirlo, deberemos trazar un plan y ponerlo en práctica.

Lo segundo es que cada jugador (humano o máquina) maneja un bando, no como en los arcades, en los que el jugador maneja a un número fijo de unidades, normalmente una. Ese bando evoluciona, creciendo en fuerza y variedad según las acciones que realice el jugador, lo cual hace que cada partida se pueda desarrollar de forma diferente, según la estrategia que siga cada jugador. A diferencia de otros tipos de juegos, tendremos libertad para modificar el entorno de juego construyendo, talando árboles, picando roca, cavando o cualquier otra forma de modificación que se nos ocurra. Normalmente, en cada bando hay distintos tipos de unidades, así como de edificaciones. Estas unidades se diferencian entre sí por la velocidad, daño que soportan, daño que infligen, capacidad de volar, etc. En general, estas características están equilibradas, de forma que una carencia de velocidad se supla con una capacidad de tiro mayor, o una velocidad alta se compensa con poco aguante al daño, etc. Para que nos hagamos una idea, cada bando tiene unidades de artillería (caras y lentas, pero duras y mortíferas)e infantería (débiles y poco dañinas, pero baratas y rápidas). Esta división puede darse tanto en unidades de tierra como en unidades voladoras, o unidades acuáticas. La división permite a cada jugador seguir estrategias diferentes, por ejemplo, basar su estrategia en la rapidez, o en la fuerza, o una combinación de ambas...

Uno de los factores más importantes de los programas de estrategia es la implementación de la inteligencia artificial

Inteligencia artificial

Lo tercero es la inteligencia artificial (IA), componente imprescindible en todo juego de estrategia, ya que el enemigo debe reaccionar a nuestros ataques de forma convincente, y atacar de formas diferentes cada vez, teniendo en cuenta nuestras defensas, a la vez que su capacidad de ataque. Aparte de las

Código de bucle principal del juego

```
LOOP
    fade_off(); //suavizamos el cambio de lo que
hubiera al menú principal
    while(fading)
        FRAME;
    end;

    //limpiamos todo bien y lo dejamos
    //bonito para el menú principal, ya
    //que no sabemos lo que nos hemos dejado

    //ahora que estamos a oscuras, lo borramos
todo y nos quedamos solos
    clear_screen();      //vaciamos la pantalla
    delete_text(all_text); // borramos todos los tex-
tos
    let_me_alone();      //el programa principal se
queda solo

    // Ponemos la pantalla de fondo que teníamos
guardada en menús.fpg
    // y ponemos los botones de las opciones.
    put(f_menus,1,0,0);

idtexto1=fun_button(f_menus,2,160,35,&butt_cre-
dits);
idtexto2=fun_button(f_menus,3,160,100,&butt_new-
game);
idtexto3=fun_button(f_menus,4,160,125,&butt_load-
game);
idtexto4=fun_button(f_menus,5,160,150,&butt_cre-
dits);
idtexto5=fun_button(f_menus,6,160,180,&butt_exit);

    // Asignamos el gráfico del ratón
    mouse.graph=1;
    mouse.file=f_mouse;
    mouse.z=-10000;

    fade_on(); //ahora que ya esta todo preparado
encendemos la luz poco a poco.

    elegida_opcion=0;
    //esperamos a que el jugador escoja alguna
opcion
    WHILE (elegida_opcion==0)
        if(butt_newgame >> 2 || key (_1))
            elegida_opcion=1; end;
        if (butt_loadgame >> 2 || key (_2))
            elegida_opcion=2; end;
        if (butt_credits >> 2 || key (_3))
            elegida_opcion=3; end;
        if (butt_exit >> 2 || key(_ESC))
            clear_screen();
            delete_text(all_text);
            let_me_alone();
            exit("Esperamos que vuelva a jugar pronto
con nosotros",0);
        end;
        FRAME;
    END

    RAND_SEED(timer*10);
    // Importante si no queremos que salgan siempre
la misma secuencia de números.
    // Inicializamos la semilla de los números aleato-
rios como el tiempo tardado en
    // escoger una opción del menú

    fade_off(); //nos ponemos a oscuras otra vez
    while(fading)
        FRAME;
    end;

    //quitamos todo lo relativo al menú
    clear_screen();
    delete_text(all_text);
    let_me_alone();

    fade_on();
    switch(elegida_opcion)
        case 1:
            put(f_menus,55,160,120);
            while(!key(_ESC)) FRAME; end;
        end
        case 2:
            put(f_menus,55,160,120);
            while(!key(_ESC)) FRAME; end;
        end
        case 3:
            put(f_menus,57,160,120);
            while ((scan_code==0) && !(mouse.right))
        FRAME; end
        end
    end//switch
END//loop
```

cuestiones bélicas, también deberá ser capaz de algo en apariencia tan simple, y sin embargo tan complejo, como el hecho de llegar a un objetivo, teniendo en cuenta que el entorno se modifica según jugamos. Es decir, debe saber escoger un camino cada vez que intenta llegar a un objetivo, ya que donde antes había un claro, ahora hay casas, o enemigos, o… Pero no os preocupéis; existen algoritmos de caminos desde la Segunda Guerra Mundial, ya que la búsqueda de caminos es de lo primero que se hizo en IA, y, cómo no, con fines bélicos. Este apartado es el más complejo ya que, de momento, sólo *deep blue* ha podido superar al ser humano en el apartado de inteligencia, y no creemos que podamos hacer un *deep blue* en un PC. En nuestro juego, este apartado se tendrá en cuenta, pero para hacerlo más fácil, en un principio haremos el juego por turnos.

En este apasionante género hay una serie de opciones fundamentales

Pasemos al juego

Lo primero que debemos decidir es la temática o al menos la ambientación del juego, y dado que en nuestro juego intentaremos usar sólo gráficos de los que vienen con DIV, la ambientación debe ser espacial, ya que es de lo que más abunda en la librería. A partir de aquí, lo más básico que debe tener cualquier juego es un menú principal, que tenga opciones del tipo «Juego Nuevo», «Cargar Juego», «Opciones», «Créditos» y «Salir del juego». Como es lo mas básico, lo codificaremos en el mismo PROGRAM, y desde aquí llamaremos a las demás funciones, como las que controlan el ratón, el juego, los botones, el sonido, etc.



La forma más sencilla de hacer un menú es considerando cada

opción como un PROCESS que tiene como gráfico el texto de la opción. Para crear estos botones, tenemos que abrir el menú de fuentes del DIV, cargar una fuente y escoger la opción «Escribir texto» de dicho menú. Escribimos «Juego Nuevo», tras lo cual nos crea un fichero de formato MAP. Seguimos estos pasos para todas las opciones, guardando los gráficos en un FPG, que llamaremos *menu.fpg*. Debemos elegir un fondo para el menú, y como nuestro juego es futurista, pondremos un fondo de planetas. El fondo lo ponemos con la función *put (fichero, gráfico, x, y)* o con *put_screen(fichero, gráfico)*. Debemos recordar que para borrar la pantalla hay que llamar a *clear_screen()*, y deberemos hacer esto cuando quitemos el menú. Una vez puesto el fondo, debemos colocar los botones correspondientes a cada opción (ver Cuadro 1). De momento, pondremos cuatro a modo de ejemplo: Juego Nuevo, Cargar Juego, Créditos y Salir. Para controlar botones usamos el proceso *fun_button (fichero, gráfico, x, y, &variable)* (ver Cuadro 2) donde *&variable* es la dirección de una variable donde se guardan los datos sobre si el ratón está encima del botón, si está apretado y qué botón está pulsado. Esta función permite controlar varios botones sin preocuparnos nada más que de las variables que guardan sus datos. Por ejemplo, si ponemos:

fun_button (f_menus, 2, 180, 80, &butt_opcion);

a partir de ese momento, la función pone el gráfico 2 del fichero *f_menus* y comprueba si se pasa con el ratón sobre ese gráfico, momento en el cual guarda en *butt_opcion* esta información. Para saber si se pulsa sobre el gráfico y con qué botón del ratón, comprobamos lo siguiente:

- IF (butt_opcion) entonces el ratón está sobre el gráfico.
- IF (butt_opcion >> 1) el ratón esta encima y tiene algún botón apretado.
- IF (butt_opcion >> 2) el botón izquierdo está pulsado.
- IF (butt_opcion >> 3) el botón del medio está pulsado.
- IF (butt_opcion >> 4) el botón derecho está pulsado.

El resultado lo podemos ver en la demo que viene en el CD-ROM de la revista.

De momento sólo es el menú, pero según avance el curso irá creciendo hasta convertirse en un juego de estrategia por turnos, o incluso puede que en tiempo real...

El ratón

Un juego de estrategia que no se maneje con ratón acabará criando polvo (virtual), en el disco duro de nuestro ordenador debido a la poca capacidad para moverse por la pantalla de las teclas. El gráfico del ratón es el que más se ve en un juego de estrategia, pero es, a la vez, el que menos debe mirarse. El ratón debe tener un gráfico original que lo relacione con el juego, pero no debe ser tan atrayente que distraiga al jugador.



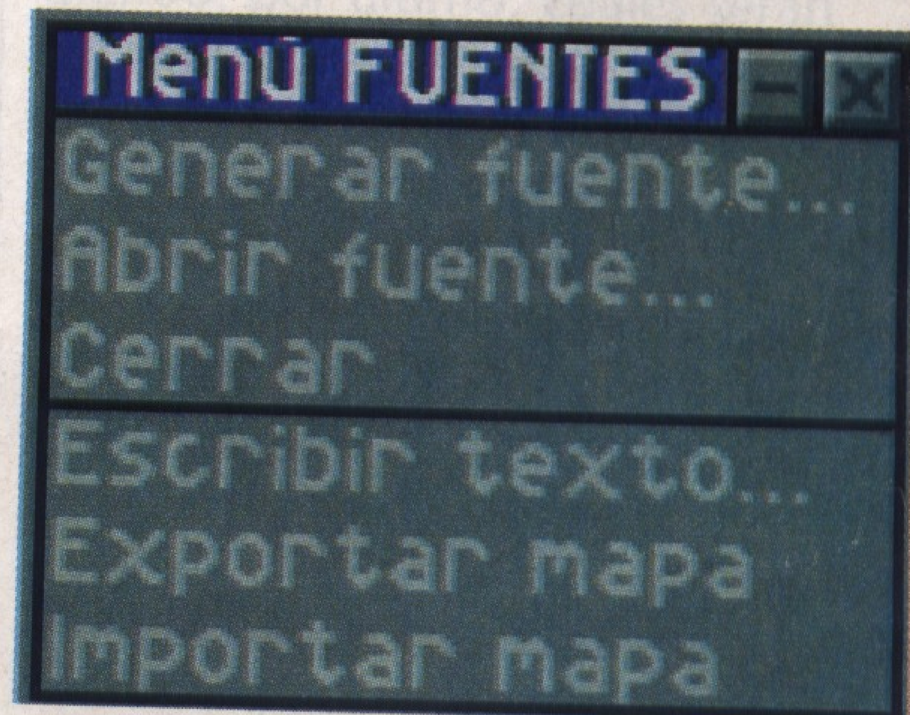
Un cursor demasiado grande puede ser muy bonito, pero estorbará al usuario, ya que no verá lo que hay debajo. En cambio, la clásica flecha blanca no le dirá nada al jugador. La forma ideal depende del argumento del juego. En un juego futurista lo adecuado puede ser una nave espacial que termine en pico. Si el juego está ambientado en la edad media, una espada podría ser un buen cursor, o una mano como tenía el Warcraft. Elijamos lo que elijamos, debe quedar claro la parte del gráfico que se usa para pinchar con el ratón. Una vez elegido el cursor principal, lo guardaremos en un FPG llamado por ejemplo *raton.fpg* ya que según avancemos en la programación veremos que necesitamos más de un cursor para el ratón.

El movimiento

El movimiento debe ser una constante en cualquier juego, ya que así parecerá más realista. Esto en un arcade no es un problema, ya que todo se mueve, pero en un juego de estrategia por turnos el movimiento no es algo intrínseco, y puede provocar aburrimiento al rato de estar jugando. El movimiento lo podemos conseguir animando el ratón, o moviendo las unidades aunque estén paradas, como si estuvieran inquietas de no hacer nada. Esto no sólo da mayor realismo al juego, sino que incita al jugador a seguir jugando. Dejar de jugar cuando no pasa nada no cuesta demasiado, pero parar en mitad de la acción es bastante más difícil. Si siempre hay movimiento, la acción se mantiene constante, pero si en la pantalla no se mueve nada, parece que el juego está en pausa.

En el menú también podemos conseguir movimiento, con alguna figura girando, con un fondo que se desplace o una opción seleccionada que tiemble.

Emilio Llamas



El menú de fuentes en DIV.

Proceso Botón

```
PROCESS fun_button (file,graph,x,y,p_variable)
PRIVATE
BEGIN
  z=-555; //no es fijo, tan sólo tiene que cumplir que el ratón esté por
encima de ellos
  priority=10000; //prioridad suficientemente alta
  LOOP
    *p_variable=0;

    if (collision(TYPE mouse))
      *p_variable += 1;
      if(mouse.left || mouse.right || mouse.middle)
        *p_variable += 2;
        if(mouse.left) *p_variable += 4; end;
        if(mouse.middle) *p_variable += 8; end;
        if(mouse.right) *p_variable += 16; end;
      end;
    end;
    FRAME;
  end;
END
```

Warcraft II

Las bases del francotirador



En esta sección nos dedicaremos a analizar lo que hacen y dejan de hacer los programadores de juegos de estrategia en éstos. En cada artículo destriparemos un juego diferente, empezando por juegos «antiguos», como el que nos ocupa, y terminando por los últimos que vayan saliendo.

Los juegos antiguos tienen menos complicaciones técnicas, lo cual nos permite un análisis de dificultad progresiva. Dividiremos la sección en un apartado de Gráficos, otro de Inteligencia y otro de Interacción.

Gráficos

Warcraft II, a diferencia de Warcraft I, usa una resolución que permite cierto «realismo» como es 640x480. La vista elegida (45º con la horizontal) da sensación de tridimensionalidad.



Gráficos necesarios para representar un Orco (la casa no).

Si nos fijamos bien, los personajes no pueden ir a cualquier punto del mapa, porque se mueven sobre un tablero imaginario, con casillas de 32x32. Para moverse de su casilla sólo pueden ir a alguna de las ocho adyacentes, y de esa a una de sus adyacentes, y así hasta llegar al punto (casilla) de destino.

Esta idea de meter las fichas sobre un tablero puede parecer muy restrictiva, pero las ventajas son más que los inconvenientes. Ya hemos visto una: reducir el número de gráficos necesarios por unidad. La siguiente ventaja es que permite hacer una función de colisión más rápida que las tradicionales: consiste en ver si las casillas adyacentes están ocupadas y por quién.

También está la técnica de pintado del mapa, generado mediante tiles. Con esto conseguimos que con sólo una matriz de 200x200 y varios gráficos de 32x32 podamos generar casi cualquier decorado de 6400x6400 píxeles. Obviamente, esto no es tan detallista como un sólo gráfico con toda la fase, pero si se hiciese así, pocos juegos de estrategia tendrían más de dos fases, cada una de 40 Mb. Además, la generación por tiles permite la creación de editores de mapas, como el que trae Warcraft II, y facilita la confección del tablero virtual.

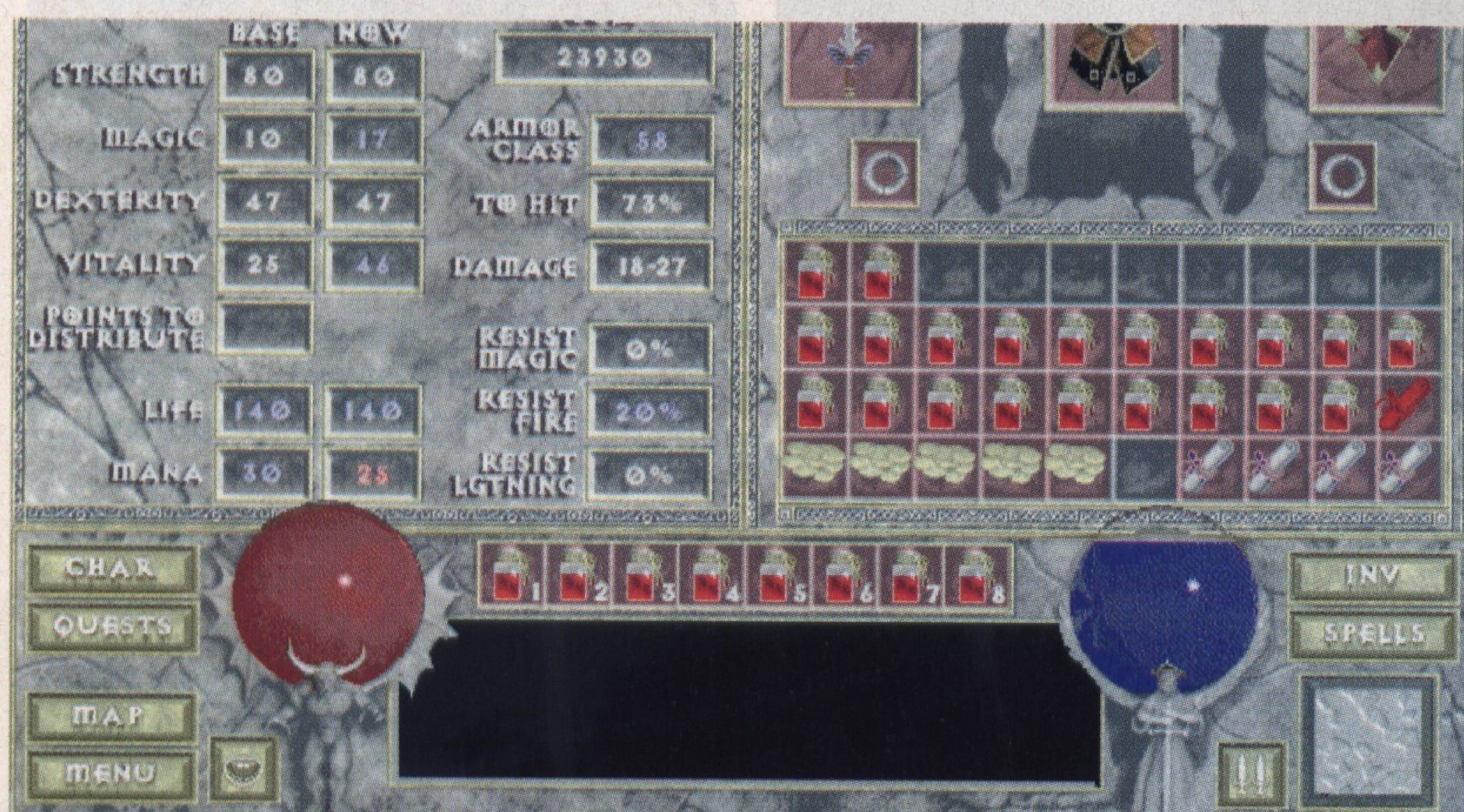
Emilio Llamas

Warcraft II y Starcraft. A diferente raza, diferente estrategia

Raza	Estrategia	Punto Fuerte
Orcos (WCII)	Ataque en oleadas apoyadas por hechizos	Rapidez y resistencia. Regeneración en los Lanzadores de Hachas.
* Zergs (Starcraft)	Ataque en oleadas sobre un terreno ya debilitado por las habilidades de algunas criaturas.	Regeneración de las unidades y el gran número de las mismas.
* Humanos (WCII)	Ataque sobre zonas concretas mediante unidades "fortalecidas" con hechizos.	Las habilidades de curación de los Paladines y los múltiples hechizos de los magos.
* Terrans (Starcraft)	Ataque táctico apoyado con unidades de espionaje y flexibilidad de las tropas para atacar o defender.	Capacidad de traslación de los edificios. Gran variedad entre las unidades que permite diferentes Tipos de estrategia.
* Protoss (Starcraft)	Ataque que se inicia con hechizos y se apoya con unidades que pueden ocultar u ocultarse.	Los escudos que se regeneran de cada unidad. Las torres defensivas Y los hechizos de los Altos Templarios.

La magia de los RPG

Aprende a crear tu propio mundo del Rol



En esta sección vamos a analizar cómo se diseña un RPG (del inglés Role Playing Game) para ordenador, siempre orientado, en lo que a programación y método de trabajo se refiere, a DIV Games Studio.

Los fundamentos de un RPG son, principalmente, el diseño de los personajes y la posterior interacción de estos en el entorno en el que los situemos. Para que tal interacción sea coherente hemos de situar la acción en un mundo o época determinado, desarrollando todo con respecto a él. Sería absurdo, por ejemplo, situar a personajes creados con el generador de personajes del Fallout (RPG postnuclear) en un mundo épico como el de Eye of the Beholder, ya que ni ser un experto con el lanzallamas ni resistir altos niveles de radiación les iba a ser de mucha ayuda en un mundo de espada y brujería. Es muy importante esta coherencia, ya que el jugador va a querer tener los mejores personajes posibles, y si no nota diferencia entre un personaje con una habilidad muy desarrollada y otro que no la tiene, sencillamente ignorará esa parte del juego. Cuanta más repercusión sobre el desarrollo del juego tenga el tipo de personaje o grupo de personajes que elija usar el jugador, mejor lo pasará explotando las habilidades de cada uno.

Un ejemplo de RPG popular es el *Advanced Dungeons & Dragons*. Sobre este juego de rol se han desarrollado muchos juegos de ordenador, y los atributos básicos que definen a cada personaje son: fuerza, destreza, constitución, sabiduría, inteligencia y carisma. Al crear un personaje se aplican bonificaciones o penalizaciones sobre estos atributos, dependiendo de la raza del personaje y su clase.

Según este esquema tenemos que un personaje se puede crear decidiendo primero a qué raza pertenece entre todas las que habiten en nuestro mundo, dándole después una profesión o clase, y decidiendo, por último, cuáles serán sus atributos básicos.

Además de unos atributos básicos podemos querer que nuestros personajes tengan a su disposición una serie de habilidades especiales. Lo importante es que cada una de estas características tenga su importancia durante el juego.

Solamente decir con respecto a esto último que el jugador de rol medio tiene bastante aguante con respecto al número de parámetros que modifican su partida, de modo que más vale pecar de complejidad que de simplicidad.

Una vez tenemos los personajes creados, hay que integrarlos en un mundo, cuidando de que las



Personajes

Atributo	Descripción
Fuerza	Poder de combate del personaje. Determina el daño infligido por sus ataques con armas de combate cuerpo a cuerpo, dagas y otras armas arrojadizas y arcos.
Constitución	Determina la resistencia general del personaje a sufrir daños. Cuanta mejor constitución tenga un personaje, menos daños recibirá en combate y más resistencia tendrá al veneno y otros modificadores de salud parecidos.
Destreza	Es el factor que mide el grado de éxito de cada acción que emprenda el personaje. Afecta tanto a la habilidad en combate como al manejo de la magia.
Magia	Poder mágico del personaje. Determina la capacidad como mago del personaje, la efectividad de los hechizos que use y todo lo relacionado con la manipulación de la magia en general.



dificultades que se vayan encontrando sean asequibles para el nivel del grupo.

El diseño del mundo abarca los escenarios de éste, dónde se desarrollará la acción, las criaturas que lo pueblan, la tecnología disponible, el clima, etc. No hay barreras para la imaginación. Las únicas barreras serán nuestras propias limitaciones a la hora de recrear el juego que tenemos en mente.

Mundo de juego

El mundo en el que se desarrollará nuestro juego será el clásico mundo de épica medieval con caballeros, magos, dragones, etc. Tendremos, por tanto, todos los elementos típicos a nuestra disposición, además de los que nuestra imaginación cree.

Generación de los personajes

Para generar los personajes que queremos introducir dentro de nuestro juego vamos a tener en cuenta una serie de atributos muy básicos, que iremos ampliando a medida que avancemos en este campo. (Si quieres centrarte en el tema puedes echar un vistazo al cuadro *Personajes*).

Bien, los que hayan jugado al programa Diablo observarán que los que aparecen en el cuadro son los atributos básicos de un personaje en dicho juego, con la diferencia fundamental de la constitución, que en el Diablo es vitalidad y modifica únicamente los puntos de vida del personaje.

Diablo es básicamente un arcade con elementos de RPG, pero a pesar de que usa pocos atributos para cada personaje los utiliza correctamente a favor de la jugabilidad. Nosotros trataremos de conseguir esto pero con más ingredientes RPG, lo cual significa más cantidad de atributos y habilidades para los personajes y mayor interactividad con el entorno.

Durante el desarrollo de todos los elementos que conformarán nuestro juego de rol iremos descubriendo cosas que habíamos pasado por alto en un principio: nuevas habilidades que servirán para nuestros personajes, nuevos monstruos para añadir a los que ya teníamos cuando nos enfrentamos al reto de hacer un RPG, nuevos tipos de armas y hechizos que no se nos habían ocurrido con anterioridad y un largo etcétera.

La idea sobre la que vamos a partir es ir introduciendo todos estos elementos sobre la marcha, pero empezando con una base sólida para no tener ningún problema durante el desarrollo, porque queremos introducir algo nuevo y no podemos porque no lo habíamos previsto. Aunque es imposible preverlo todo, sí se puede crear una base sólida sobre la que será fácil añadir nuevos conceptos y ampliar los ya existentes. Hemos de pensar en un RPG en su forma más amplia.

Las habilidades que definiremos en principio serán las que aparecen dentro del cuadro *Habilidades* que aparece en esta misma página.

Tendremos en cuenta una serie de razas que son comunes a la mayor parte de los mundos de fantasía épica. (Para comprobarlo, asómate al cuadro que analiza todas las *Razas comunes*).

Esta división de razas es propia de nuestro juego, y aunque en algunos casos habrá detalles comunes con otros juegos y la historia en general de la fantasía épica, en otros muchos crearemos detalles nuevos dentro de este universo. Y es ahí donde entra en juego la imaginación de cada uno.

Habilidades

Habilidad	Descripción
Combate con espadas	Proporciona al personaje una bonificación en la lucha con el arma en cuestión. Esta bonificación se refleja en la posibilidad de acertar un ataque y en el daño infligido por éste.
Combate con hachas	
Combate con mazas/martillos	
Combate con arcos	
Combate con armas arrojadizas	
Magia del Fuego	Determina el nivel de habilidad de un personaje para manipular la magia elemental correspondiente. Según este nivel de habilidad, el personaje podrá aprender hechizos más complicados dentro de esa disciplina elemental, y utilizará de forma más efectiva los que ya conozca de ella.
Magia del Aire	
Magia de la Tierra	
Magia del Agua	

Razas comunes

Raza	Descripción
Humano	Son buenos guerreros. Tienen bonificación en fuerza.
Elfo	Son buenos ladrones. Tienen bonificación en destreza.
Enano	Son buenos guerreros. Tienen bonificación en constitución.

Sonido para tus creaciones

Aprenderemos a trabajar con las ondas

Uno de los aspectos más importantes en la confección de un juego es todo lo referente al sonido del mismo. Una espectacular banda sonora y unos efectos de calidad pueden ser unos ingredientes ideales para conseguir el efecto deseado en las personas que hagan uso de nuestros programas.

En esta sección vamos a analizar una herramienta que puede ayudarnos mucho a la hora de dotar a nuestras creaciones de sonido. El programa en cuestión es Cool Edit 96, un editor de samples para Windows 95 y Windows NT, y su utilidad es, básicamente, la de grabar y reproducir ficheros de audio e÷n una amplia gama de formatos, así como la de editarlos, mezclarlos entre sí y convertirlos de un formato a otro.

Cool Edit 96 es una de las mejores interfaces que hay en estos momentos para realizar trabajos con sonidos de todo tipo

La interfaz de Cool Edit 96

Cuando ejecutamos el programa, observamos los siguientes elementos (de arriba abajo):

- Una barra de menú con la que podremos acceder a las diferentes opciones del programa.
- Una serie de iconos con motivos mnemotécnicos, que dan acceso a las opciones de uso más común.
- La porción de la pantalla donde se muestra la onda.
- Botones de reproducción, grabación, parada, zoom,etcétera.

Para empezar a trabajar necesitaremos cargar un fichero de sonido desde disco, o bien obtenerlo de una fuente de audio externa mediante el sampler de nuestra tarjeta de sonido. Grabar nuestra propia voz mediante un micrófono nos proporcionaría un buen ejemplo de fichero de sonido sobre el cual poder realizar nuestras pruebas.



La interfaz de Cool Edit 96. Podemos acceder rápidamente a las opciones más útiles del programa mediante los iconos que aparecen sobre la onda.



Podemos trabajar simultáneamente con varias ondas gracias a la función New Instance. Con esta opción facilitaremos nuestro trabajo en las mezclas de audio.

Entrando en materia

En la opción *File* de la barra de menú disponemos de todas las operaciones usuales de manejo de ficheros. Además se incluye una opción bastante interesante, que nos permite manejar varios samples al mismo tiempo, la función *New Instance.* Esto puede resultar útil en el caso de querer copiar partes de un sample en otro. Basta con abrir dos sesiones de *Cool Edit*, una con cada sonido, y copiar el segundo a continuación del primero en la sesión donde teníamos cargado este último (el bombo). Después, bastará con reproducir el sample en modo *loop.*

Si pinchamos con el ratón en la opción *View* accedemos a todas las funciones referentes a la visualización de los samples. Estos se pueden presentar de dos maneras: en la forma de onda (*Waveform View*) o bien en su forma espectral (*Spectral View*). Esta última nos presenta la onda con gran colorido, pero, aunque su apariencia sea más atractiva, es menos práctica que la primera. Desde *View* también se puede acceder a las siguientes funciones:

- Visualización de tan sólo uno de los canales, el izquierdo o el derecho, en caso de que nuestro fichero de sonido sea estéreo, es decir, que esté compuesto por dos ondas.
- Se puede especificar si se quiere visualizar la onda completa o sólo una parte, mediante la función *Viewing Range.* Así mismo, podremos elegir en qué unidades deberá expresarse el tiempo de muestreo de la onda (función *Display Time Format*), y la escala de frecuencias que aparece en la parte derecha de la interfaz del programa (*Vertical Scale*).
- Por último, podemos acceder a una serie de funciones más accesorias, tales como información referente al sample, listas de reproducción de samples e incluso un reproductor de CD.

Experimentando con los Samples

En la opción *Transform* de la barra de menú es donde encontramos todo un universo de posibilidades para, como dice la misma palabra, transformar nuestros samples.

Ahora entramos en el campo del retoque de sonidos. Las primeras opciones relacionadas con este campo las encontramos en el menú *Amplitude.* En este menú encontramos las siguientes opciones:

- *Amplify*: Esta opción nos permite amplificar el volumen de una onda hasta un 300%. Tiene algunos modos predefinidos de amplificación que nos permiten, por ejemplo, realizar

efectos de balance en los ficheros estéreo. Son las llamadas panorámicas en los estudios de grabación musical.

- *Channel Mixer*: Esta opción permite modificar por separado determinados atributos de los canales izquierdo y derecho de un fichero estéreo, tales como el volumen o su situación respecto a su eje horizontal. Así mismo, permite mezclar ambos canales en uno sólo, dando mayor peso a uno de ellos.
- *Envelope*: Permite modificar la envolvente de una onda. Posiblemente sea un efecto demasiado sutil como para que resulte de interés.
- *Normalize*: En el mundo de la música este efecto sería equiparable al llamado *compresor*, y lo que se consigue con él es limitar el volumen de la onda, para que no exceda de un valor dado, de forma que se eviten subidas estridentes.

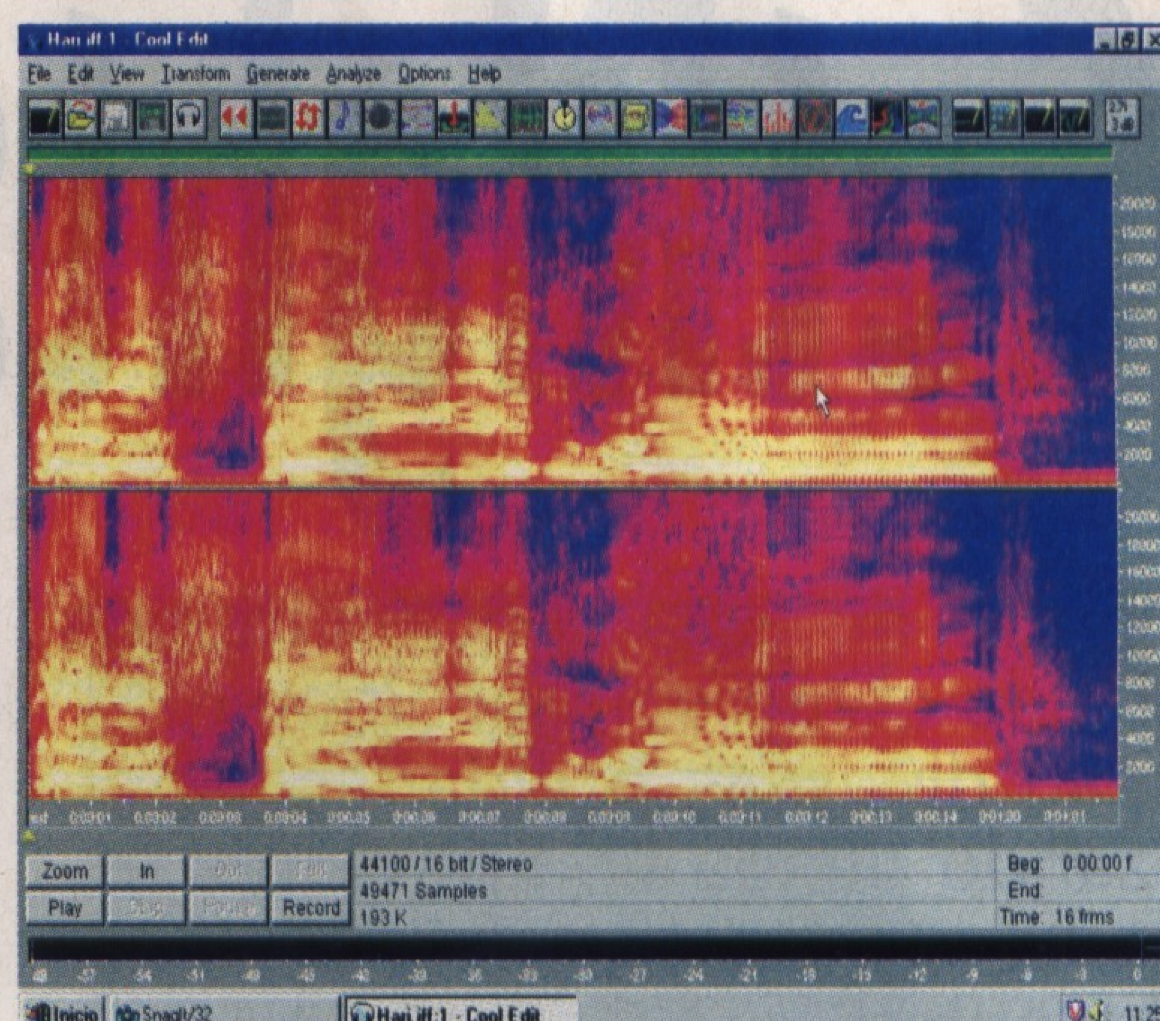
Es posible aplicar filtros (función *Filters*), que nos permitirá potenciar determinadas frecuencias de nuestros samples, obteniendo mayores niveles de graves, medios o agudos.

Esos espectaculares efectos

Ahora vamos a pasar a analizar uno de los puntos más interesantes de Cool Edit 96, sus potentes efectos. Si accedemos al menú *Delay Effects*, tendremos acceso a los siguientes efectos:

- *Delay*: Retarda el comienzo de la onda. Suena poco espectacular, pero mezcla la onda original con otras versiones de la misma a las que hayamos aplicado el *Delay* y cambios de volumen. El efecto es similar al eco, pero es más versátil, ya que se puede hacer el efecto contrario, anticipando el sonido a volúmenes inferiores, como si la fuente sonora estuviera alejada y se fuera acercando a nosotros paulatinamente (por ejemplo, un coche que se acerca y luego se aleja, partiendo sólo del sonido del motor a un volumen constante).
- *Echo*: Consigues eco, pudiendo seleccionar el número de rebotes por unidad de tiempo (*ratio*), su retardo (*delay*) y el volumen inicial de los mismos. Hay ecos predefinidos que nos ayudan a conseguir ecos característicos de algunos lugares.
- *Echo Chamber*: Este efecto es similar al anterior, pero en este caso los parámetros que se introducen son tales como el tamaño de la habitación donde se supone que se quiere ubicar el sonido, o el lugar exacto donde se colocó el micrófono.
- *Flanger*: Este es uno de los efectos de uso más extendido con guitarras eléctricas. Lo que hace es modificar paulatinamente la ecualización del sample a lo largo del mismo, consiguiendo un curioso efecto futurista. También se dispone de algunas selecciones predefinidas, que resultan ideales para modificar la voz del usuario y hacerla sonar como si de un extraterrestre se tratase.
- *Reverb*: Utilizado en todas las grabaciones de los cantantes del panorama musical desde el comienzo de los tiempos. Es un eco corto y suave que trata de imitar la reverberación de los sonidos en la naturaleza, y que, dicho sea de paso, tapa bastantes fallos de los divos. Se trata del efecto perfecto para aquel que se anime a incluirse cantando en sus creaciones.

Accediendo al menú *Special* de la opción *Transform* de la barra de menú, se nos presentan tres atractivas opciones: La primera es la de crear música con la capacidad de ayudar a nuestro cerebro a relajarse (*Brainwave Synchronizer*). Podría resultar interesante en el caso de padecer algún tipo de desorden nervioso, pero no es una de las opciones más prácticas a la hora de poner sonido a nuestros juegos de ordenador. La segunda opción nos permite poner *Distorsión* a nuestros samples, lo cual es útil para hacer que las voces suenen como si se estuviera hablando por radio, o para samplear nuestra maltrecha guitarra española y hacer que suene como si estuviese en las manos del mismísimo Van Halen. También incluye una serie de configuraciones predefinidas que nos ayudarán a conseguir mejores resultados.


Visualización de la onda en su forma espectral.

La tercera opción, *Music*, es especialmente atractiva. Gracias a ella podemos acceder a un pentagrama en el cual podremos escribir una composición musical que será interpretada con el sonido que estemos editando en ese momento.

La última opción del menú *Transform* es *Time/Pitch* que nos permitirá modificar el periodo de tono y el *tempo* de nuestros samples sin que uno afecte al otro, si así lo deseamos. Podremos reproducir una frase en un tono mucho más agudo, sin que varíe el tiempo de reproducción de la misma, o bien reproducir una frase más despacio sin que ésta suene más grave de lo normal.

La creación de nuevos efectos de sonido es uno de los mayores atractivos de los editores

Sonidos de la nada

En la opción *Generate* de la barra de menú se nos ofrecen tres opciones que nos permitirán crear sonidos sin haber cargado previamente ningún fichero de audio:

- *DTMF Signals*: Se introduce un número telefónico y *Cool Edit* nos devuelve los tonos que se producirían al marcar el citado número.
- *Noise*: Sirve para crear ruido. Útil en caso de necesitar una ventisca improvisada.
- *Tones*: Nos permite crear tonos de una determinada frecuencia. Podríamos generar una nota musical (o un acorde) y luego utlizarla con la opción *Music*, ya comentada anteriormente, para componer una canción.

Sergio Cánovas

Clasificación de las funciones de Cool Edit 96

Retoque	Efectos	Creación
Amplify	Delay	Music
Channel Mixer	Echo	DTMF signals
Envelope	Echo Chamber	Noise
Normalyze (Compresor)	Flanger	Tones
Filtros		Reverb
Noise Reduction	Distorsión	
	Time/Pitch	

GetRight: Un download

Dentro de esta sección iremos viendo cómo son y cómo se usan diversas utilidades, analizando desde compresores hasta editores hexadecimales. Describimos en este artículo el uso de GetRight, un programa que nos evitará muchos problemas.

Aunque la mayoría de los navegadores tienen incorporada una utilidad para copiar ficheros de forma remota (*download*), estas utilidades suelen carecer de la opción de parar y continuar a partir del punto en que lo dejamos, por lo que tenemos que empezar de cero si se corta la conexión.

Esta utilidad es imprescindible para optimizar al máximo nuestros tiempos de conexión a la Red de redes.

Con un poco de imaginación, nos damos cuenta de lo molesto que puede ser llevar grabado un 95% de un fichero después de dos horas y que se corte la conexión. Pues esta situación no es tan rara, sobre todo si se tiene un módem lento. Para evitarnos tener que empezar de nuevo y esperar otras dos horas, con la consiguiente pérdida de tiempo y dinero, tenemos GetRight.

Con este programa podremos parar la conexión cuando deseemos (y como deseemos: apagar el ordenador, desenchufar el módem, etc.) y volver a empezar desde donde estábamos sin ningún tipo de pérdida.

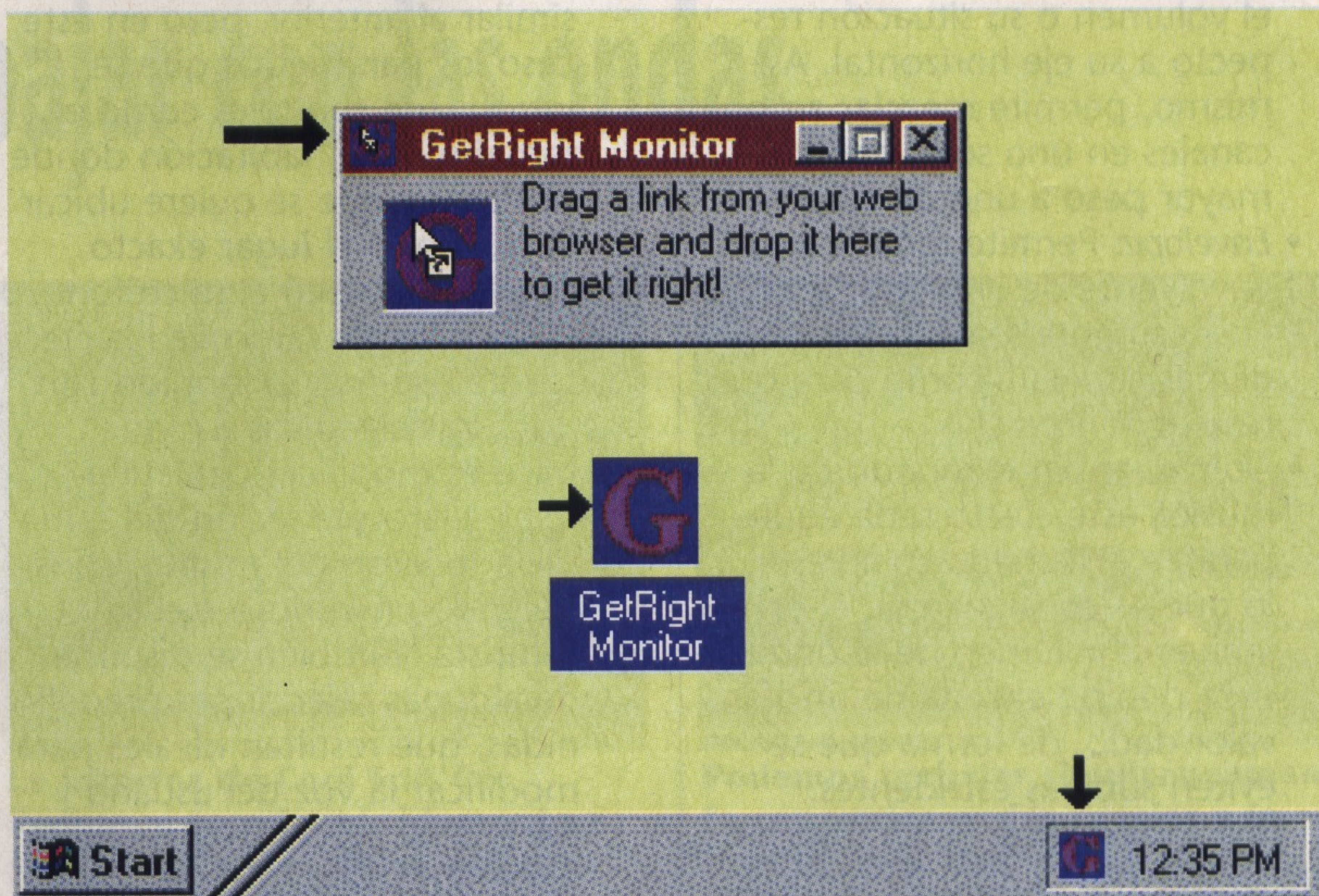


Figura 1. Aquí vemos dónde pulsar para abrir el menú emergente.

Instalación y Configuración

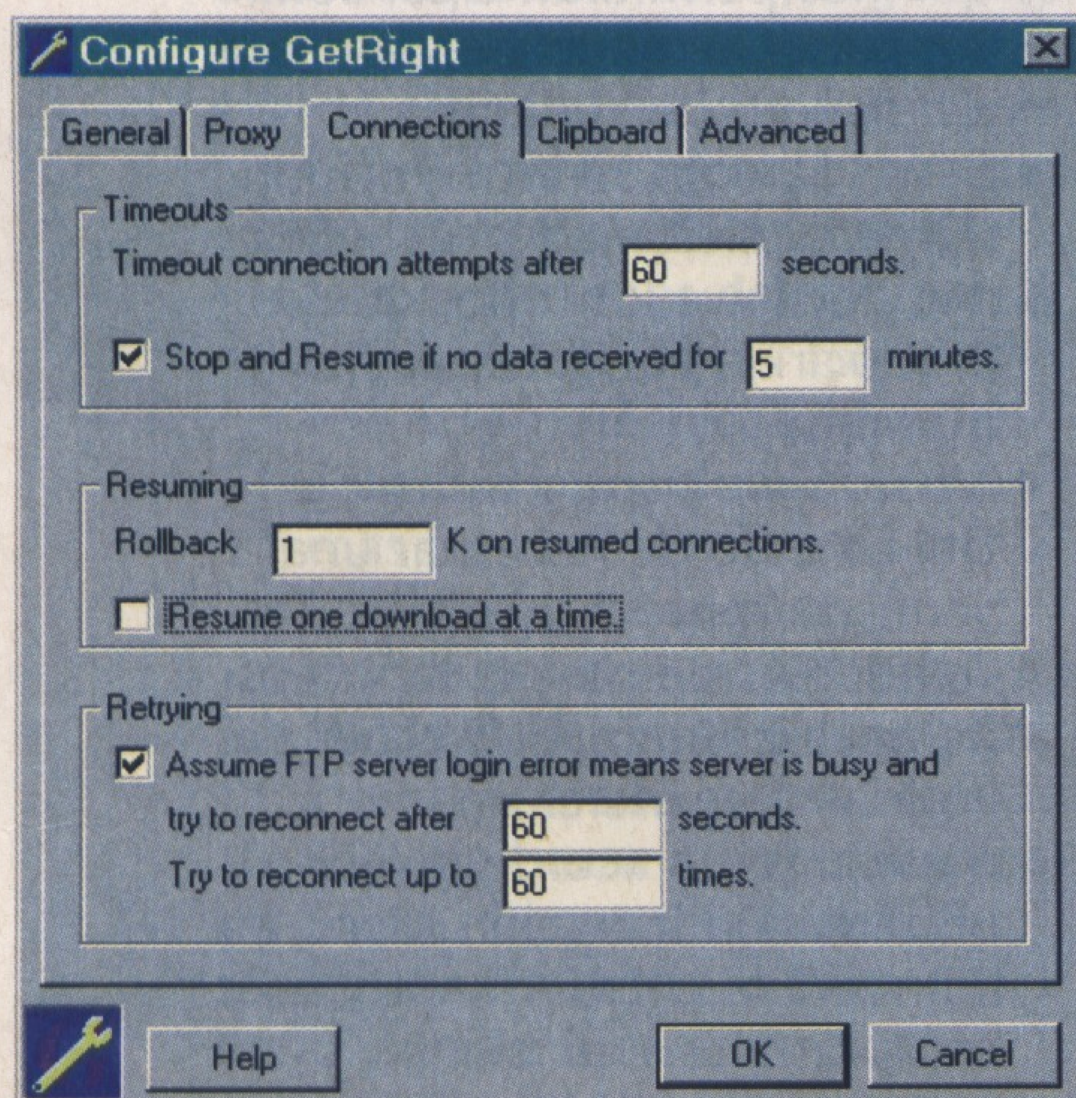
Lo primero que debemos hacer es instalarlo, para lo cual tendremos que seguir las indicaciones que nos irá dando el programa de instalación. Nos pedirá que elijamos el directorio donde instalar el programa. Si nos conectamos con frecuencia, es aconsejable crear un acceso directo en la carpeta Inicio, para que se ejecute el monitor de GetRight cuando arranquemos el ordenador.

Una vez instalado tenemos que configurarlo, para lo cual podemos hacer dos cosas:

- Ejecutar desde la carpeta de GetRight el programa de configuración.
- Pulsar en el monitor de GetRight con el botón derecho del ratón y seleccionar la opción *Configure...* (ver fig. 1 para saber dónde pulsar).

Una vez en el menú de configuración tenemos cuatro submenús: *General, Proxy, Connections, Clipboard y Advanced.*

En el submenú *General* configuramos qué queremos que pase cuando se termine de «bajar» con éxito un fichero o cuando se corte la conexión, además de nuestra dirección de correo electrónico (sólo es necesaria para ponerla como clave en los servidores FTP anónimos. Es una norma de cortesía de la red), y el código que nos dan al registrarnos. Si no nos registramos, aparece una ventanita cada vez que usamos el programa recordándonos que debemos hacerlo para seguir usándolo. En el submenú *Proxy* configuramos los datos de nuestro servidor *proxy*, si lo tenemos. No es necesario rellenar nada en este submenú, aunque, si se dispone de un *proxy* su uso es aconsejable, ya que acelera sensiblemente el proceso de copiado. Los datos del *proxy* debe facilitárnoslos nuestro servidor de acceso a Internet. En el submenú *Connections* configuraremos una de las cosas mas importantes: el *Rollback*. Éste indica cuántos Kbytes retrocedemos (ignoramos) de lo bajado en la última conexión interrumpida, es decir, ignoramos los últimos Kbytes de lo que tenemos de fichero, empezando a coger un poco por detrás de donde lo dejamos. A mayor *Rollback*, más seguro es el proceso y más tiempo se pierde. Un *Rollback* pequeño significa menos tiempo perdido, pero menos seguro el «empalme» de lo que teníamos con lo que cogemos ahora. Cada uno deberá configurarlo según sus prioridades. También se configuran los *Timeouts* es decir, tiempos tras los cuales se realiza una determinada acción, o se deja de realizar. En el submenú *Clipboard* elegimos los tipos de ficheros que se empiezan a «bajar» sin pedir confir-



Aspecto del menú de configuración.

inteligente

mación al usuario, y los tipos que GetRight no aceptará copiar. Estos últimos podemos hacer que sean enviados a un *browser* del tipo que definamos y que ya esté ejecutándose cuando intentemos copiarlo. Por último, en el submenú *Advanced* elegimos detalles como, por ejemplo, si queremos que al minimizar el monitor de GetRight, este se minimice en la barra de tareas al lado del reloj, o lo haga de forma normal. También configuramos las rutas de:

El directorio donde GetRight guardará por defecto los archivos «bajados».

Un antivirus para pasárselo a los ficheros que copiemos.

El archivo de *log*.

A estas alturas del artículo ya deberíamos tener configurado el GetRight... Y ahora, ¿qué?

Cómo empezar a grabar un archivo

Tenemos cuatro opciones para empezar a grabar un archivo de la Web, así que no tenemos excusa para no usar GetRight:

Portapapeles, el más fácil de usar del los cuatro métodos:

GetRight controla constantemente lo que se copia al portapapeles, y si es una dirección de Internet automáticamente empieza a copiarlo en nuestro disco duro.

La forma de copiar un enlace en el portapapeles es la siguiente:

Si usamos un navegador de Netscape, «pincharemos» con el botón derecho del ratón sobre el enlace, seleccionando después la opción *Copy link location* en el menú emergente.

Si usamos un navegador de Microsoft (para Windows 95/NT), «pincharemos» con el botón derecho del ratón sobre el enlace, seleccionando después la opción *Copy shortcut* en el menú emergente.

Si usamos un navegador de Microsoft (para Windows 3.x), «pincharemos» con el botón derecho del ratón sobre el enlace, escogiendo después la opción *Propiedades* en el menú. Una vez en este menú, seleccionamos el texto que hay en el recuadro etiquetado como URL y pulsamos Ctrl-C para copiarlo al portapapeles.

Desde cualquier programa (correo, procesador de textos, etc.), seleccionamos la dirección del archivo y pulsamos Ctrl-C para copiarlo al portapapeles.

Si no funciona, tal vez es que el tipo de archivo que queremos bajar está anulado en el submenú de configuración *Clipboard.* Podemos, aún así, iniciar la copia de forma manual, o bien configurar de nuevo las opciones del submenú *Clipboard.*

Drag-and-Drop, la forma clásica de copiar ficheros con GetRight:

Desde un navegador que soporte *drag-and-drop* podemos pinchar sobre el *link* sin soltar el botón del ratón y arrastrarlo hasta la ventana de GetRight, donde soltaremos el botón.

Una vez hecho esto, GetRight nos preguntará si queremos que empiece a copiar el fichero.

¡Cuidado! Si intentamos arrastrar un *link* sobre el icono minimizado de la barra de tareas (los que aparecen al lado del reloj) no servirá de nada. Debemos hacer doble clic con el botón izquierdo del ratón sobre el icono para que se abra la ventana del monitor GetRight para una vez abierta, arrastrar el *link* sobre la ventana. Si nos inclinamos a usar esta opción, es aconsejable marcar en el menú emergente (ver Figura 2) que la ventana esté *Always on top* ya que así podremos verla con el navegador maximizado.

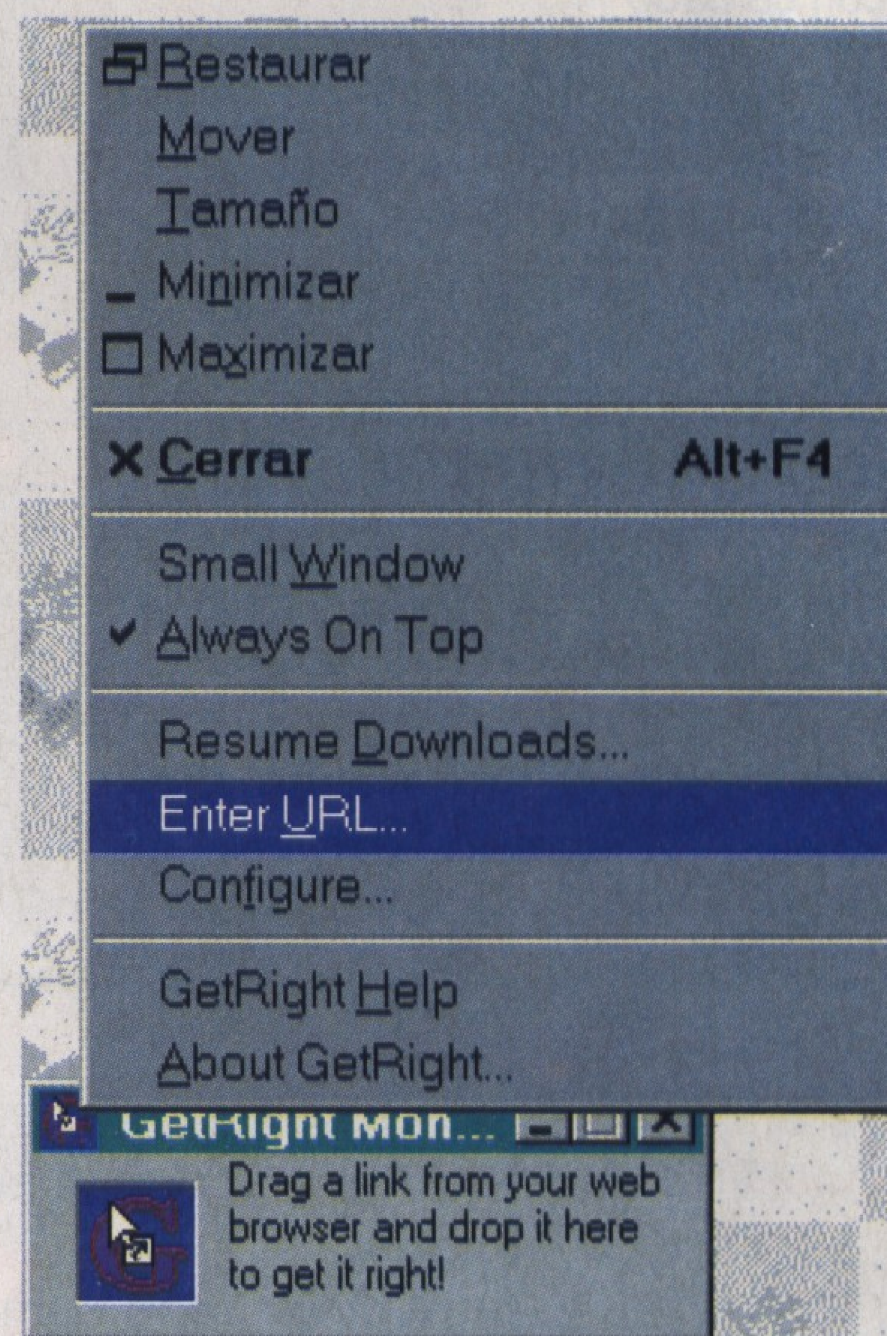


Figura 2. Este es el aspecto que tiene el menú emergente bajo Windows 95.

Escribiendo la dirección: (ver figura 2).

Seleccionamos la opción *Enter URL* del menú emergente.

Escribimos la dirección en el cuadro, o pulsamos Ctrl-V para pegarla desde el portapapeles.

Pulsamos Aceptar y GetRight nos preguntará si queremos copiarlo y en caso afirmativo, con que nombre.

Para usuarios avanzados:

GetRight puede ser ejecutado seleccionando ejecutar en el menú

Glosario de Términos

HTTP: *Hyper Text Transfer Protocol*
Protocolo para la transferencia de hipertexto. Un ejemplo de hipertexto son las páginas .html que se publican en la www.
FTP: *File Transfer Protocol* Protocolo para la transferencia de ficheros.
Es un protocolo dedicado exclusivamente al copiado de ficheros de forma remota, por lo que las transferencias que hagamos desde un servidor FTP serán más seguras y rápidas que las que hagamos desde uno HTTP.
Download: Copiar de forma remota un fichero. («bajar»)
URL: Dirección de un fichero en Internet. Suele tener la forma:
Protocolo://nombre.servidor/ruta.archivo/nombre.archivo
Proxy: Un servidor *proxy* es un puente entre nuestro ordenador e Internet. Normalmente se comparte entre muchos usuarios, y va guardando las páginas a las que cada usuario accede y los ficheros que se «baja». Así, si alguien ha accedido antes a lo que nosotros queremos, nos ahorramos conectar con otro servidor. Nos lo manda nuestro *proxy*.

inicio, o desde una ventana DOS, y pasándole como parámetro la dirección del fichero a «bajarnos».

Por ejemplo: GetRight *http://www.dominio.com/ruta.fichero/nombre.fichero*

Será muy sencillo manejar esta herramienta si seguimos con cuidado todos los pasos.

Ahora que ya sabemos cómo empezar a grabar, no está de más saber cómo seguir con una sesión interrumpida (voluntaria o involuntariamente), ya que esta es la principal ventaja de GetRight.

Cómo continuar con lo que esta a medias

Para continuar copiando un fichero, lo mas importante es que el servidor permita esta acción, o tendremos que empezar de cero. La mayoría de los servidores FTP permiten continuar copiando. En cambio, no todos los servidores HTTP permiten esta opción. La mayoría de las veces GetRight detecta automáticamente al inicio de la sesión si el servidor nos permite o no «resumir», para que podamos buscar un servidor que lo permita. De todas formas, si vamos a «bajarnos» un fichero muy grande, conviene comprobar manualmente si el servidor permite «resumir» o no. Para hacerlo, pausamos el copiado a los pocos Kbytes copiados, para luego «resumir» la sesión. Si GetRight sigue donde lo paramos, entonces el servidor permite continuar. En cambio, si empieza de nuevo, lo mejor que podemos hacer es rezar para que no se corte la conexión como hacíamos con los programas antiguos.

Ahora que sabemos si podemos o no continuar, vamos a ver cómo «resumir». Tenemos varias opciones: Si la ventana de GetRight para el fichero está todavía abierta, tan sólo deberemos pulsar el botón *Resume*.

Si la ventana ya está cerrada, deberemos llamar al programa GetRight – *Resume All Requests* que comenzará a copiar todos los archivos que tengamos a medias.

También podemos seleccionar la opción *Resume Downloads...* del menú emergente (ver figura 2) donde seleccionaremos qué ficheros «bajar» de los que nos quedan.

Ahora ya sabemos «resumir». Finalmente, comentar que para desinstalar GetRight sólo hay que ejecutar el programa *Uninstall GetRight* que se instala con todo lo demás.

Para terminar

Para más información sobre GetRight, los datos de *HeadLight Software* son:

e-Mail: *mjb@oneworld.owt.com*

Web: *http://www.headlightsw.com/*

Correo: PO *Box 514 Richland*, WA 99352 (USA)

Hasta el próximo numero de la revista. Esperamos que este haya sido útil.

Emilio Llamas

Drag-and-Drop

El *Drag-and-Drop* es una opción muy común en programas para el entorno Windows.

Consiste en manejar todo como objetos, es decir, que podemos coger un fichero y meterlo en una carpeta de forma literal: «pinchamos» en el objeto sin soltar (cogemos), llevamos el ratón hasta ponerlo sobre la carpeta donde queremos dejar el archivo, y soltamos (metemos el archivo en la carpeta). Esto permite un manejo más intuitivo de las aplicaciones, ya que nos permite manejar los datos de la forma en que lo haríamos en la vida real.



¿Quieres ganar grandes premios por tus juegos hechos en DIV? ¿Quieres que tus juegos salgan comentados en tu revista favorita?
Para ello, mándanos tus trabajos realizados en DIV, y podrás ganar 25.000 ptas. al primer puesto, y 20.000 ptas. a los segundos puestos.

E-Mail. divmania@prensatecnica.com
Concurso juegos DIV Manía

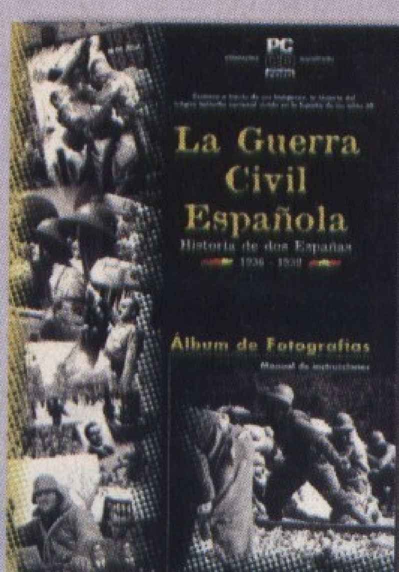
C/Alfonso Gómez, 42, nave 1-1-2
C.P. 28037
Madrid (España)

- La editorial se reserva los derechos de difusión de los juegos ganadores, (siempre y únicamente las versiones enviadas).
- No se devolverán los juegos enviados.
- La lista de nuevos ganadores se irá publicando en cada número de DIV Manía.



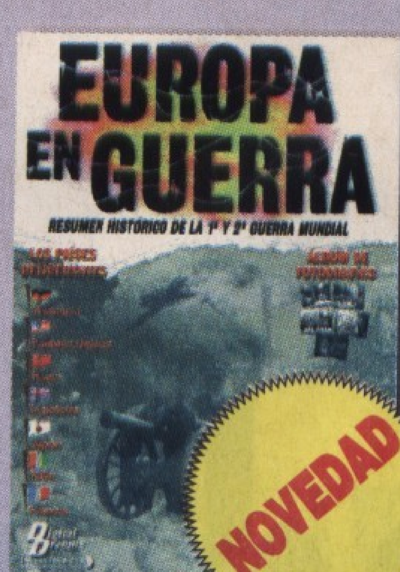
Una obra imprescindible en la biblioteca de cualquier aficionado al fútbol y a la historia. Le permitirá revivir a través de datos e imágenes, los mejores momentos de la actual Champions League.

2.995 ptas.
Incluye CD-ROM.



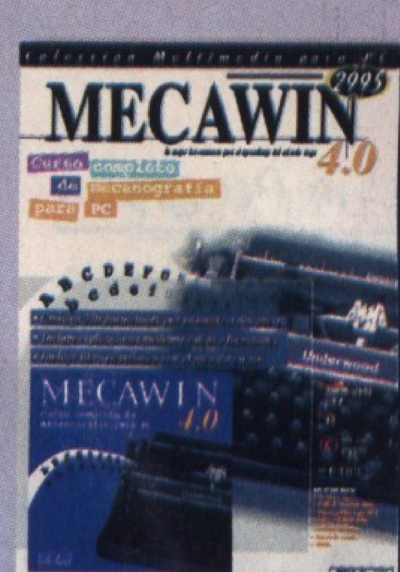
Esta publicación electrónica tiene como objetivo dar a conocer los motivos, acontecimientos y consecuencias de la Guerra Civil Española vivida en los años 30, analizando todos los aspectos.

3.995 ptas.
Incluye CD-ROM.



El presente trabajo de investigación da a conocer al usuario el desarrollo de la Primera y Segunda guerra mundial. Contiene fotografías, vídeos, voces y se regala un libro de 192 págs. con el CD.

3.995 ptas.
Incluye CD-ROM.



Curso completo de mecanografía que ofrece la posibilidad de establecer una base firme para noveles, así como una perfecta corrección de "manías" de escritura para los más avanzados.

2.995 ptas.
Incluye CD-ROM.



CD-Rom que recoge alrededor de 700 recetas de cocina tradicionales. Cuenta con cálculo energético de calorías, valores nutritivos de cada alimento, etc.

1.995 ptas.
Incluye CD-ROM.



Programa que ayuda al usuario a llevar un seguimiento regular de las jornadas para poder realizar quinielas con el mayor acierto posible. Una herramienta muy útil para los aficionados al fútbol.

1.995 ptas.
Incluye CD-ROM.



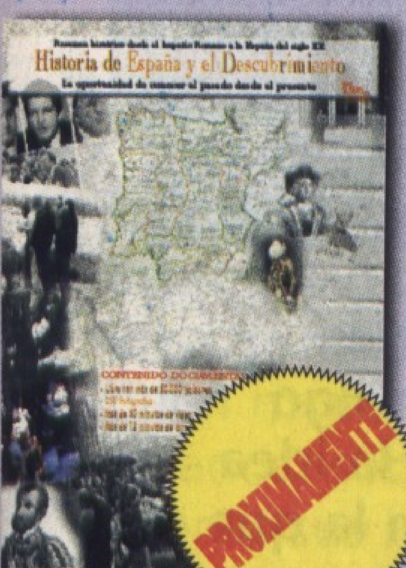
Fotografías e ilustraciones explicativas a lo largo de todos los capítulos del temario. Animaciones en 3D sobre mecánica y diversas maniobras con vehículos.

2.495 ptas.
Incluye CD-ROM.



Incluye preguntas de las oposiciones más importantes del Estado. Entre ellas se encuentra el MIR, Administración del Estado, Auxiliar Administrativo, Administración del Inem.

2.495 ptas.
Incluye CD-ROM.



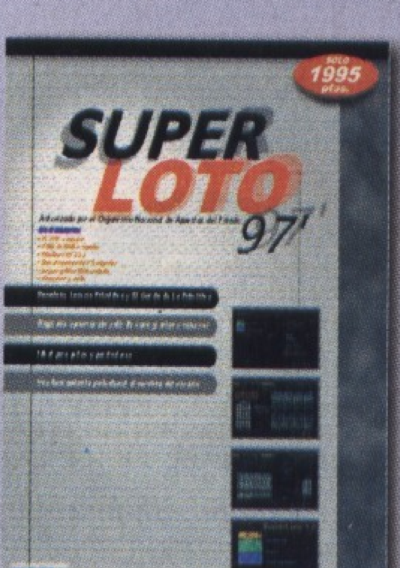
Sumerge en el tiempo al usuario visitando cualquier momento clave desde la prehistoria hasta la transición. Juego de preguntas y respuestas, opciones de búsqueda, 250 fotografías, 35 minutos de vídeo en movimiento y narraciones sonoras...

2.995 ptas.
Incluye CD-ROM.



Con variadas novedades y mejoras entre ellas, Iberconta 2.0 con la incorporación del módulo de profesionales (Estimación objetiva y Directa) e Iberfacturación 2.0, que permite la impresión de facturas sin límite de páginas.

2.995 ptas.
Incluye CD-ROM.



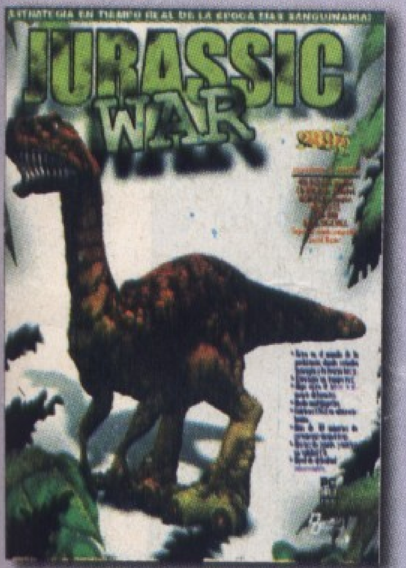
SuperLoto 1.0 es una herramienta pensada para los aficionados a los juegos del 6-49, Lotería Primitiva, Bonoloto y El Gordo de la Primitiva. Es un programa informático para uso doméstico, oficinas de apuestas y peñas.

1.995 ptas.
Incluye CD-ROM.



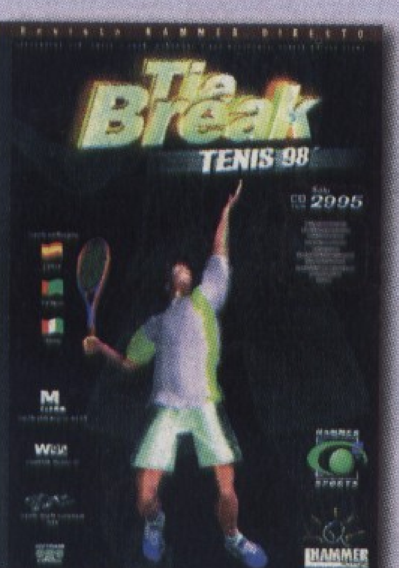
Artes Marciales recoge todo lo relativo al origen, práctica, técnicas y personajes de este deporte. Incluye un apartado dedicado únicamente a la defensa personal, donde se enseña a defenderse ante situaciones de violencia.

2.995 ptas.
Incluye CD-ROM.



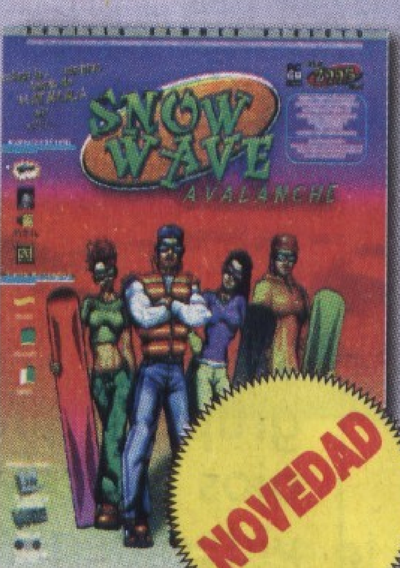
Jurassic War es el primer juego de estrategia en tiempo real que te conduce a la prehistoria. Elige tu tribu y lucha por el dominio de la isla en una época donde la emoción y los peligros acechan en cualquier momento.

2.995 ptas.
Incluye CD-ROM.



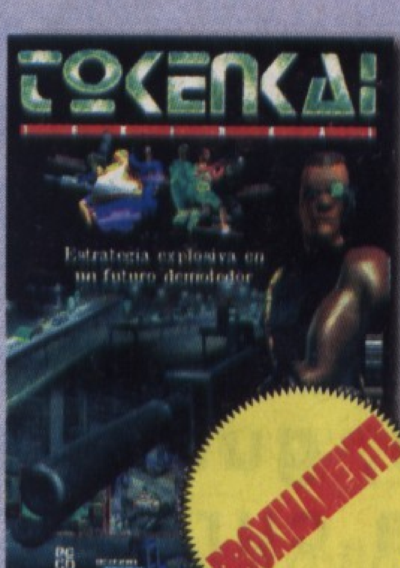
El juego de tenis más espectacular. El simulador 3D definitivo de tenis. Con increíbles efectos visuales como destellos de sol, sombras dinámicas, fuentes de luz y cámaras móviles que seguirán la acción en todo momento.

2.995 ptas.
Incluye CD-ROM.



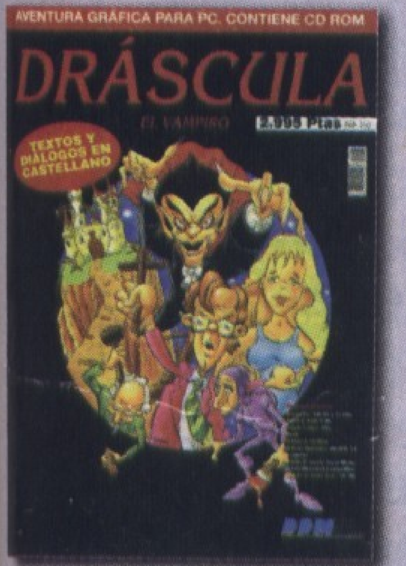
Disfruta con Snow Wave Avalanche de vertiginosos descensos entre riscos y cañones helados, realizando aterradores saltos al vacío e imposibles acrobacias dignas de un auténtico campeón.

2.995 ptas.
Incluye CD-ROM.



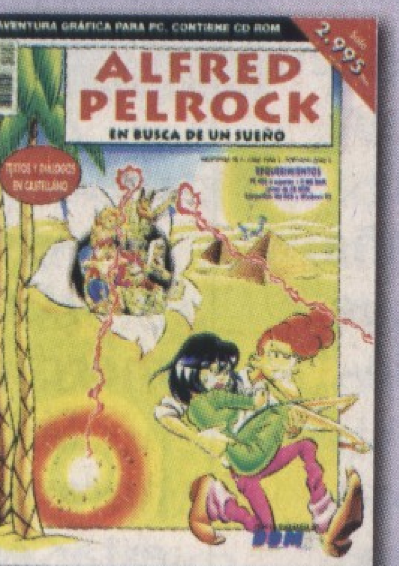
Tokenkai nos sumerge en una sociedad futurista gobernada por los intereses económicos donde la pobreza y el crimen imperan en las calles. Tan sólo eliminando al líder, Rojas, conseguirás tu objetivo: acabar con el cartel de narcotráfico más grande del mundo.

2.995 ptas.
Incluye CD-ROM.



Aventura gráfica que se desarrolla en Transilvania. El protagonista comienza allí su viaje, durante el cual se enamora de una chica. Los problemas surgen cuando el Conde Dráscula secuestra a la joven.

2.995 ptas.
Incluye CD-ROM.



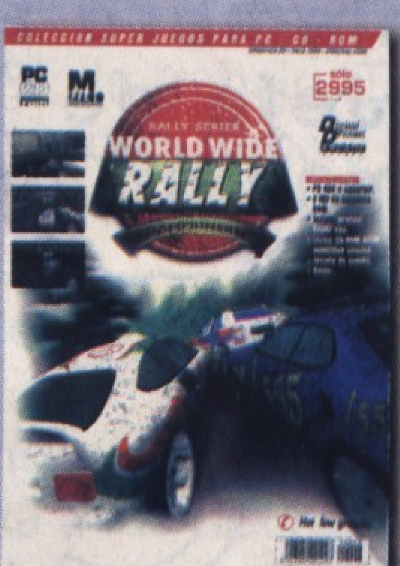
Aventura gráfica con numerosas animaciones y efectos especiales. La acción se desarrolla en el Antiguo Egipto, cuando una princesa es castigada por su padre a vivir en una pirámide.

2.995 ptas.
Incluye CD-ROM.



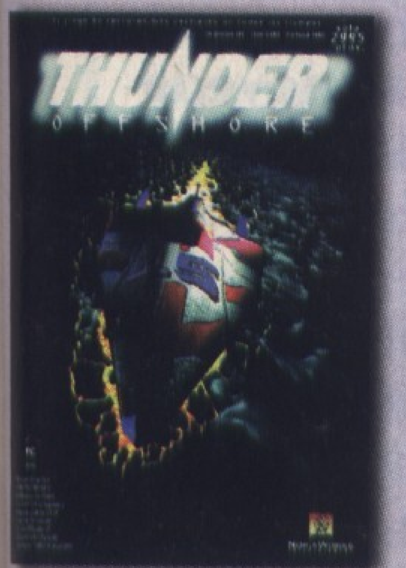
Entretenido programa ideado para los aficionados a los juegos de tapete. Combina elementos muy atractivos: un juego de cartas y posibilidad de participar con bellas señoritas que irán desprendiéndose de su ropa.

1.995 ptas.
Incluye CD-ROM.



Un arcade de carreras en 3D que hará las delicias de todos los *fitipaldis*. Gracias a su alta resolución, a su soporte multijugador y a sus 15 cuidadas pistas podrás sentirte un auténtico piloto al volante de tu coche preferido.

2.995 ptas.
Incluye CD-ROM.



Modelos 3D voxelizados de geometrías superiores a 50 mil polígonos, paisajes fotorrealistas con resolución automática y superficies reflectantes gracias a la técnica VTP. 3 Equipos, 6 thunder Arrows y 15 circuitos diferentes.

2.995 ptas.
Incluye CD-ROM.



Permite crear juegos comerciales y libres de royalties. Posee un entorno integrado que incluye un diseñador gráfico, generador de fuentes de letras y explosiones, así como 15 juegos y multitud de tutoriales.

4.995 ptas.
Incluye CD-ROM.



Contiene ocho de los mejores juegos de todos los géneros existentes: Arcade, Aventura, Deportivo, Simulador... Una colección imprescindible repartida en 2 CD-Rom.

2.995 ptas.
Incluye 2 CD-ROM.

Digital Dreams multimedia

Solicite su ejemplar enviando este cupón por correo, por Fax: (91) 304.17.97 o llamando al teléfono (91) 304.06.22 de 9:00 a 19:00h.

Si, deseo realizar el siguiente pedido:

LTIMEDIA Y UTILIDADES

- AMPEONES DE EUROPA 2.995
- A GUERRA CIVIL ESPAÑOLA 3.995
- UROPA EN GUERRA 3.995
- ECAWIN 4.0 .... 2.995
- UE VIANDAS 2.495
- UPER QUINIELAS 4.0 .... 1.995
- UTOR MULTIMEDIA DE AUTOESCUELAS 2.495
- UIA DEL OPOSITOR 2.495

- ❑ HISTORIA DE ESPAÑA Y EL DESCUBRIMIENTO 2.995
- ❑ IBERGESTION 2.995
- ❑ SUPER LOTO 1.0 .... 1.995
- ❑ ARTES MARCIALES 2.995

VIDEOJUEGOS

- ❑ JURASSIC WAR 2.995
- ❑ TIE BREAK TENIS 2.995
- ❑ SNOW WAVE AVALANCHE 2.995
- ❑ TOKENKAI 2.995
- ❑ DRÁSCULA 2.995
- ❑ ALFRED PELROCK 2.995
- ❑ STRIP POKER 1.995
- ❑ WORLD WIDE RALLY 2.995
- ❑ THUNDER 2.995
- ❑ DIV GAMES STUDIO 4.995
- ❑ TOTAL GAMES 2.995

mbre y apellidos .... Domicilio .... Población ....
vincia .... CP .... E-mail .... DNI/NIF ....

RMA DE PAGO

❑ Talón a DIGITAL DREAMS MULTIMEDIA ❑ Contra-reembolso
❑ Giro postal n° .... de fecha ....
❑ Tarjeta de crédito ❑ VISA n° ❑ AMERICAN EXPRESS n° ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
❑ Fecha de caducidad de la tarjeta .... Nombre del titular, si es distinto ....

Firma,

Rellena este cupón y envíalo a:
DIGITAL DREAMS MULTIMEDIA
C/ Alfonso Gómez,42 Nave 1-1-2
28037 Madrid.

Programadores en acción

Compañías que buscan un hueco en el mercado

Hace diez años nació Balance, una productora nacional de videojuegos. Ha sido un trayecto corto, pero intenso a juzgar por los frutos que ha dado, con conocidos juegos de calidad (quién no recuerda el soberbio Fire Wind), apreciados tanto dentro como fuera de nuestras fronteras. Nos hemos interesado por sus actuales proyectos, así que, guiados por nuestra curiosidad, entrevistamos a Antonio Ruiz, miembro fundador y actual Jefe de Programación del grupo.

Compañía:	Balance
Juego en desarrollo:	Sin titular
Género:	Carreras de motos
Título previo:	Fire Wind

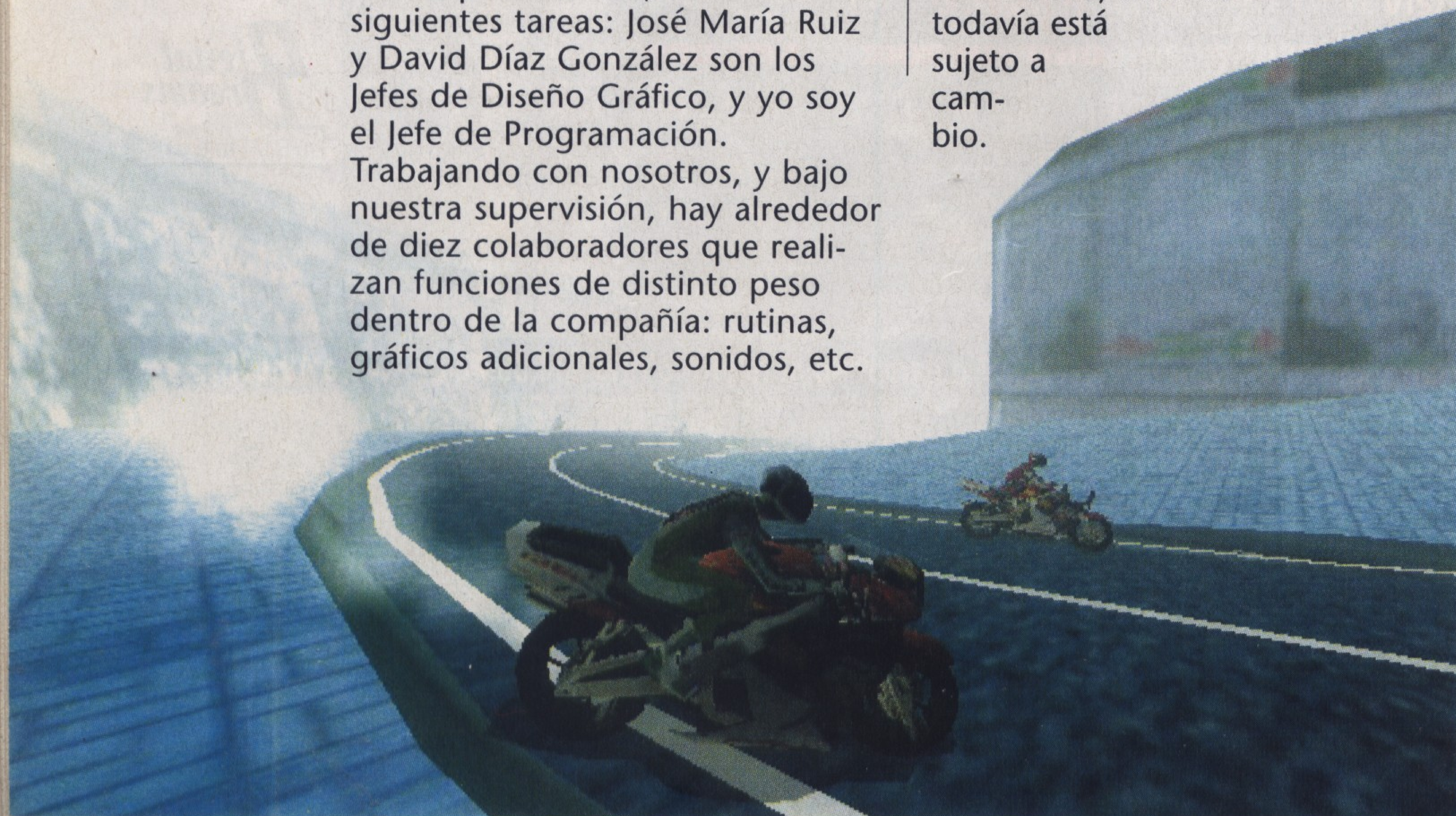
Bal@nce

¿Cuántas personas integran el equipo de Balance? Somos tres miembros fundadores, que además de gestionar la empresa y de ser los productores, realizamos las siguientes tareas: José María Ruiz y David Díaz González son los Jefes de Diseño Gráfico, y yo soy el Jefe de Programación. Trabajando con nosotros, y bajo nuestra supervisión, hay alrededor de diez colaboradores que realizan funciones de distinto peso dentro de la compañía: rutinas, gráficos adicionales, sonidos, etc.

No siempre la estructura de esta compañía estuvo tan clara. En sus comienzos tuvieron algunos problemas con un grupo de daneses que trabajaban conjuntamente con el grupo y que no veían con buenos ojos la actividad de desarrollar videojuegos. Esta confrontación dio lugar, por iniciativa danesa, a la desintegración del grupo. Surgió así uno nuevo que, teniendo en cuenta la organización actual, podríamos denominar "primario", afincado en España, y un segundo, y secundario, que siguió su curso de forma independiente.

¿Cuál es el nombre del proyecto que tenéis actualmente entre manos?

Aún no podemos hablar de su título, además todavía está sujeto a cambio.



¿Nos podrías comentar algo acerca de la idea original en la que se basa?

Se trata de hacer un juego de simulación de motos que compita directamente con los grandes del género. Estamos utilizando las tecnologías más avanzadas, de modo que estará especialmente orientado a los usuarios de tarjetas aceleradoras 3Dfx, sin olvidar una opción de aceleración desde el propio software.

¿Cuándo se inició el presente proyecto?

En Junio del año pasado.

¿Cuál es su estado actual de desarrollo?

Aproximadamente el 85%.

¿Cuándo esperáis que salga al mercado?

Si todo se desarrolla según el horario previsto, el juego debería estar en la calle estas Navidades o, a más tardar, el primer mes del año que viene.

Comenta un poco las novedades de las que dispondrá este proyecto referentes a los gráficos y la programación.

Sobre todo unos gráficos 3D de gran detalle y múltiples efectos de iluminación, aparte de la ya habitual opción de juego en red. Siempre sin olvidar una elevada jugabilidad.

¿En qué plataformas va a funcionar?

En PC.

¿Podrías hablarnos un poco acerca de los futuros proyectos de la compañía, en caso de que existan?

Sí, existen futuros proyectos. Pero, de momento, no te puedo comentar nada al respecto, ya que tenemos un contrato de confidencialidad firmado con Dinamic.





Conclusión

Estas palabras nos pueden dar una idea muy aproximada acerca del buen momento que atraviesa Balance, sobre todo si tenemos en cuenta sus humildes orígenes: un grupo de aficionados a los videojuegos y a la programación de demos en los ordenadores de 8 y 16 bits. Todos sus primeros juegos, no comercializados, fueron desarrollados (qué tiempos) para los aquel entonces todopoderosos Amiga500 y Amiga1200.

El crecimiento de la compañía vendrá acompañado con un aumento de la plantilla. De hecho, si alguno de vosotros visita la página que tienen en la red.

Ignacio Pulido



Compañía:	Hammer Tech.
Juego en desarrollo:	Hellfire
Género:	Acción
Título anterior:	Snow Wave



En el presente texto se comenta otro de los programas que se encuentra en desarrollo en nuestro país, de la mano de uno de los desarrolladores, Rubén. Se trata de un título de Hammer en el que tendrás que manejar un potente helicóptero.

Número de integrantes
Aunque el numero de integrantes varía, básicamente el proyecto está siendo desarrollado por 2 programadores, 3 grafistas y 1 coordinador.

Idea original del proyecto.
El proyecto nace de una idea que teniamos en mente ya hace mucho tiempo, pensamos en que no existía ningún producto con una temática similar al nuestro que satisfaciese al usuario en todos los aspectos.

Fecha prevista de salida al mercado.
Algo importante que he aprendido a lo largo de mi carrera es que en el mundo de los videojuegos las fechas no existen, o si existen a nosotros no nos gustan. Por eso somos partidarios de que, aunque planeemos las fechas de salida, el juego no verá la luz hasta que el desarrollo nos satisfaga al 100%, aunque provoque un retraso.

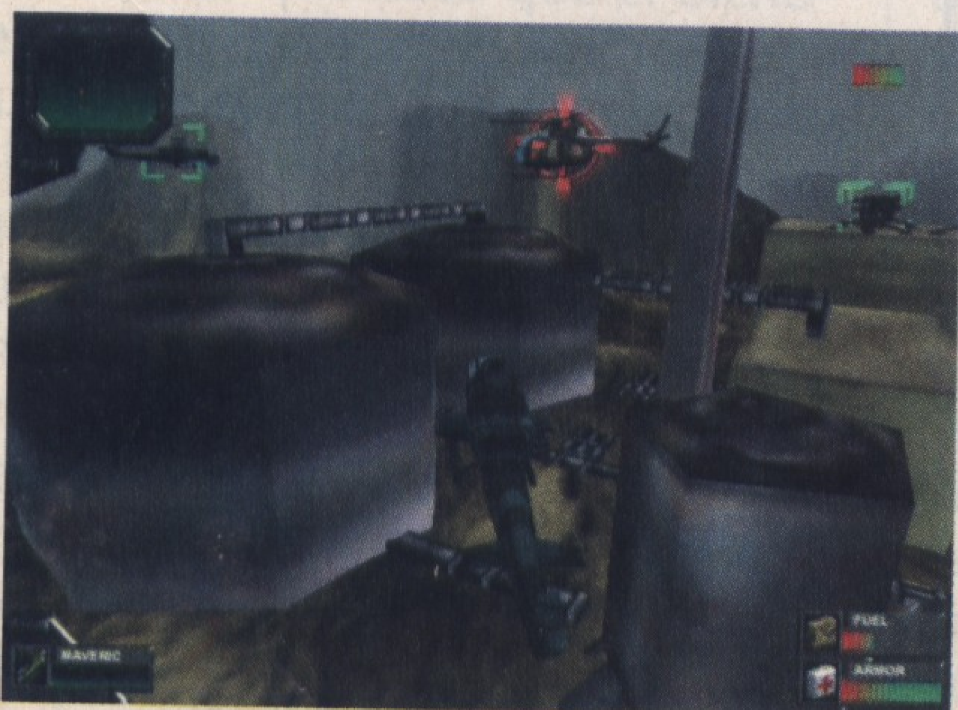
Tiempo de desarrollo total y parcial, es decir, cuánto se lleva hasta ahora.
El tiempo estimado de desarrollo es de 11 meses y actualmente llevamos 3 meses intensivos de desarrollo, sin tener en cuenta el Engine3D que ha sido desarrollado previamente.

Novedades a nivel de gráficos y programación que tiene el proyecto.
Las novedades técnicas que incluirá el juego es algo que preferimos reservarnos y que vosotros mismos podréis comprobar a su debido tiempo.

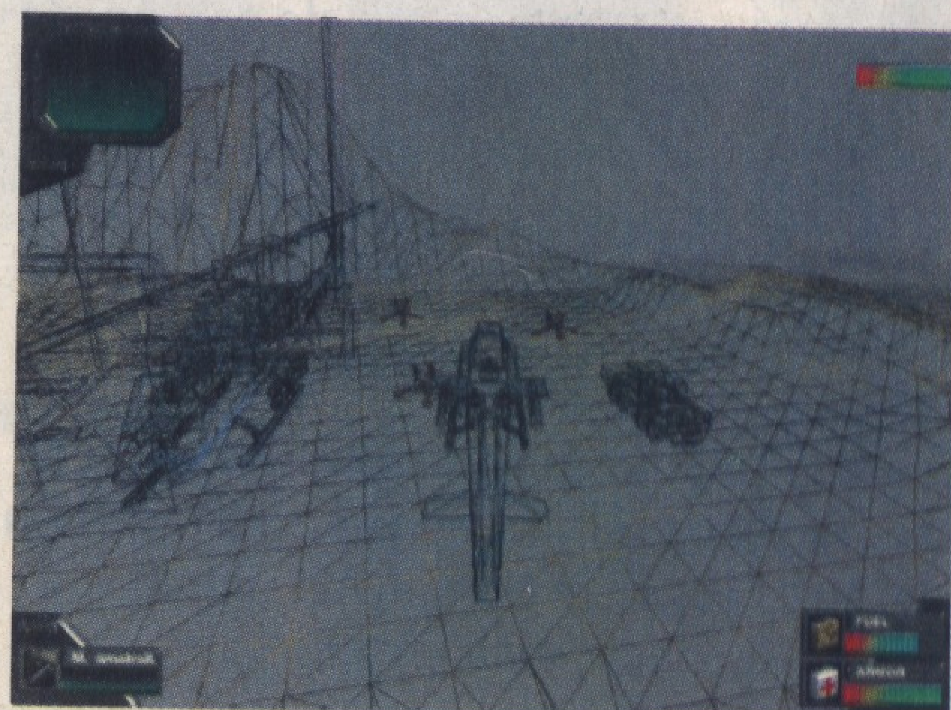
Estado actual del desarrollo.
El proyecto podríamos decir que acaba de nacer, porque actualmente está a un 25 % de trabajo. Todavía nos queda un arduo trabajo para alcanzar nuestro objetivo.

Plataformas en las que funcionará el proyecto.
Nosotros somos un equipo que tenemos en cuenta a los distintos tipos de usuarios que tenemos en el mercado, con lo cual, a pesar de que programamos juegos que debido a su potencial requieren altas máquinas, no olvidamos a aquellos usuarios que no disponen de grandes equipos, permitiendo regular la calidad del juego y que éste pueda correr en máquinas inferiores.

Futuros proyectos, si los hay.
Esto es algo que prefiero reservarme para el futuro.







El correo del lector

La voz de los lectores es escuchada y atendida

En un mundo como el del compilador DIV, en el que la única razón verdadera que nos mueve a seguir avanzando y a lanzar revistas como la presente es el usuario y el lector, hay que prestar atención a todo lo que podáis decirnos, ya se trate de críticas o de aportaciones.

En busca de scroll

Estoy haciendo un programa, pero no consigo poner un scroll de fondo. El fondo por ahora es de un mapa, y quiero que se mueva cuando un muñeco se encuentre en una posición determinada por «x» en la pantalla a la misma velocidad del muñeco: *x+=12*.

NICK: Rojie
E-mail: *miorba@ctv.es*
¿Cómo puedo hacer para que un disparo desaparezca cuando choque con un enemigo?

La respuesta es que donde esté el proceso que carga al enemigo, hay que meter dentro de la cadena la siguiente orden:
if(collision(type [nombre de el proceso de el disparo]))
break;
end;
Esta orden suele introducirse dentro de un until o un bucle.

NICK: Rojie

Un grafista para el tenis

Somos un par de amigos que estamos desarrollando un juego de tenis. Necesitamos un grafista 2D para los menús del juego. Enviad muestras de vuestro trabajo y referencias de los programas que usáis.

NICK: RCZ
E-mail: *rcz@arrakis.es*



DIV en la red

La presente carta ha sido parcialmente reproducida en otra sección de esta misma revista, pero su interés para todos los lectores que no dispongan de conexión a Internet y no conozcan lo que pasa en ese especial mundillo nos impulsa a mostrarla de nuevo a todos ellos.
Estimados colegas, *DIVadictos* y *DIVas* de todo tipo (si las hay, que no se dejan ver), para todos los que estéis interesados en desarrollar el juego compartido de la ciudad ideal que propuse, os comento lo siguiente:
- Estoy desarrollando una página web con el contenido de la propuesta más detallado, con vuestras ideas para ayudar, planteamientos, código, etc. pero he de deciros que todavía tardará unos pocos días en aparecer. Estoy esperando las vacaciones para ello, ya que el *kurre* y otros proyectos en DIV no me dan más tiempo, de momento, pero tranquilos, que todo llegará (esto es como el DIV PRO, vamos)
- Todavía necesito más apoyo de vuestra parte ¿o es que no os gustaría cambiar algo en el lugar en el que vivís? La propuesta es común y los interesados deberíais dejaros ver algo más. ¡Yo organizaré vuestras propuestas y serán mostradas en la futura web! Debéis de tener en cuenta que la ciudad se ha de crear desde 0, sin *impeDIVmentos*, ¿ok?
- Para mis colegas *chateros*: algunas semanas no podré dejarme ver por el chat *but the job!*, así que utilizad el *e-milio* que es casi gratis gracias o *for de job* Telefónica. Nada más por el momento. Volveréis a saber de mí.
!Deemo

Nick: !Deemo
E-mail: *ptomas@avl.es*
Hola *again*!
La semana *ke* viene paso por BCN, *¿ké* tal una *kedada* por allí?
¿Y en el resto de España? ¿Sólo va a haber *kedadas* en la Jamonería Salamanca?

!Deemo
Nick: !Deemo
E-mail:*ptomas@avl.es*

El correo, un apartado imprescindible

Esta sección denominada Correo, intentará ser una especie de tablón de anuncios o foro de todos los usuarios de DIV y lectores de esta revista. Por lo tanto, esta sección necesita de la participación de todos, con sus anuncios, dudas, respuestas, etc. Es más, nosotros, como coordinadores de la revista, únicamente nos haremos eco de vuestras opiniones. Siendo también vosotros los que contestéis dichas dudas. Es decir, esta sección será una especie de tablón de anuncios, donde se podrán hacer ofertas y demandas de cualquier tipo, como pueden ser grafistas, programadores, etc. Además de ser también un foro de preguntas y respuestas. En los dos casos, seréis vosotros quienes responderéis a los mensajes mes a mes. Esperamos vuestras cartas y mensajes. Por último, comentar que no se censurará ningun mensaje, por lo que se pide corrección en el lenguaje.

Un "matamarcianos"

Soy DM.
Aunque es la primera vez que escribo, ya hace tiempo que estoy apuntado. Un saludo a todo el mundo. Ahora estoy con un jueguecillo de navecillas en scroll vertical.
Esto va para los dibujantes o semejantes: necesito que alguien me indique cómo dibujar alguna nave, las mías me salen ridículas y sin vida (si alguien quiere ser bueno y enviarme algún gráfico, ya sabéis donde estoy).

Nick: DAVID-CARME
E-mail:*DAVID.CARME@infomail.lacaixa.es*



Un truquillo

Hola. He de decir que ¡por fin he podido hacer que las balas sean instantáneas! Al igual que para hacer, por ejemplo, que un proceso que busque un camino sea inmediato, lo que se consigue suprimiendo el frame.
O si por ejemplo se quiere que una nave se mueva más rápido que otra y no se quiere utilizar el avanzar o las coordenadas (x,y) para que no haga saltos. Utiliza una función como *if(rand(0,1)==1) frame*
Es decir, unas veces hará el frame (para que el proceso no se bloquee) y otras no, para que acelere. Espero que esto le sirva a alguien. Truquillos como éstos son bienvenidos.

nightwol@teleline.es
 - Nightwolf
Nick: Nightwolf
E-mail:NIGHTWOL@teleline.es

Una buena solución

Para Hengomar:
Se me ocurre una manera, pero no la he probado y no sé si funcionará, porque no conozco muy bien el lenguaje DIV (soy un poco novato) pero tengo experiencia en Pascal, Ensamblador, etc. y así es como lo haría si lo hiciera en estos lenguajes. Bueno, ahí va:
imaginemos que tenemos un procedimiento que hace que el bicho se agache, salte, etc., después de que comprueba las teclas para esto y lo hace (pone en pantalla) saltamos a otro procedimiento que hace los movimientos especiales. En este procedimiento tenemos una variable global, por ejemplo de texto con 8 caracteres ASCII. Aquí vamos guardando la ultima tecla pulsada del procedimiento anterior en orden acumulativo, sin machacar ninguna posición (mientras que no se llenen los 8 huecos, en este momento habría que correr todos los movimientos una posición hacia la izquierda). En este procedimiento también hay un temporizador que si no se ha pasado ninguna tecla del procedimiento madre (no se ha hecho ningún movimiento) borra esta variable ASCII, y la siguiente vez que se le pase tecla empezaría con la variable vacía. Luego sólo queda ir comprobando las últimas teclas pulsadas y ver si coinciden con alguna configuración de movimientos guardados, y si es así mover al bicho como sea.
Bueno, espero que te sea de ayuda y espero que funcione
Un saludo a todos los de la lista.

Oscar Perea

Nick: Oscar Perea Marcos
E-mail: *toolkit@batch-pc.es*

Movimientos especiales

¿Sabe alguien cómo hacer para que los personajes de un juego ejecuten una acción determinada tras un movimiento especial? *Es decir, después de hacer una combinación tipo* Street Fighter II:*zabajo,abajo+adelante,adelante+puñetazo=onda* *(por ejemplo).*
Llevo varias semanas intentándolo y aunque creo que estoy a punto de lograrlo no lo doy por hecho. Hasta estos momentos los personajes sí responden a la combinación anterior, pero cuando sólo pulso abajo éstos no se agachan. Y cuando corrijo este error dejan de responder al movimiento especial.
¿Tiene alguien una rutina ya hecha o sabe cómo hacerla?
Cogido de la lista del Canal DIV.
Nick: "Hengomar"
E-mail: *hengomar@meridian.es*

Esta cuestión ha salido muchas veces en el canal, éste es un programa de ejemplo que hice, algunos ya lo tenéis. Como no voy a estar localizable durante cierto tiempo, si os surgen dudas, preguntadlas en el canal y, posiblemente alguno de los que discutimos este programa os responda.

Nick: Sevi
E-mail: *<jmsevicom@apdo.com>*

Esperamos vuestros comentarios

La dirección de correo para enviar vuestras ofertas, demandas, preguntas o respuestas, es la siguiente:

Revista Divmanía
Sección Correo
C/ Alfonso Gómez, 42 nave 1-1-2
28037 Madrid (España)

También podréis usar la siguiente dirección de correo electrónico, indicando en el "Subject", como tema, "DIVmania - Seccion: DIV-me". Todos los "chateros" ya tendréis noticias nuestras a estas alturas.

divmania@prensatecnica.com

No queremos marcharnos sin animaros a todos a que sigáis escribiendo *mails* o, directamente, cartas para esta sección. Podéis mandaros mensajes unos a otros y utilizar estas páginas como contacto, siempre y cuando los temas que tengáis que tratar tengan alguna relación con nuestra pasión: la programación de videojuegos y, sobre todo, la herramienta que nos la hace más fácil: DIV Games Studio. Como podéis comprobar por las notas y misivas que hemos publicado en este número, la "censura" que nos permitimos en esta parte de la revista es mínima. Sin caer en la vulgaridad, se respetarán los textos originales. Dentro de dos meses volveremos a ponernos en contacto en una sección que muy pronto será vuestra preferida.

Antonio Marchal



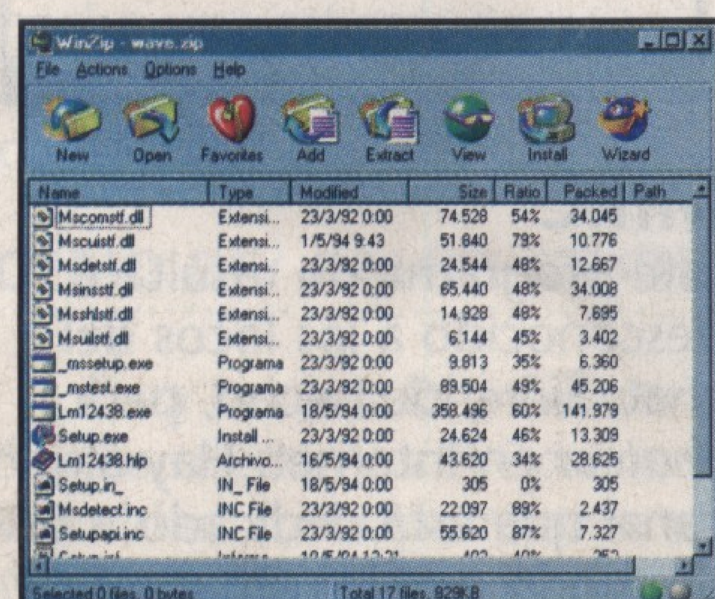
Permite la conversión de gráficos y en las últimas versiones permite ver animaciones e incluso oir sonidos. Este programa es imprescindible para cualquier grafista.

VIRUSSCAN DE MCAFEE 3.1.8.

Es uno de los antivirus más extendidos. Es capaz de detectar y limpiar una gran cantidad de virus. También puede trabajar residente en memoria y detectar cualquier virus que entre en el sistema por cualquier medio, ya sea por disco, Internet o red local, VShield se activa y avisa de que hay un virus.

WINZIP 7

El compresor de archivos más conocido. Permite trabajar con diferentes formatos como ARJ, LZH, ARC, TAR, TGZ, Unix Compress, UUEncode, XXencode, BinHex, MIME y, cómo no, ZIP.



LIBRERÍAS

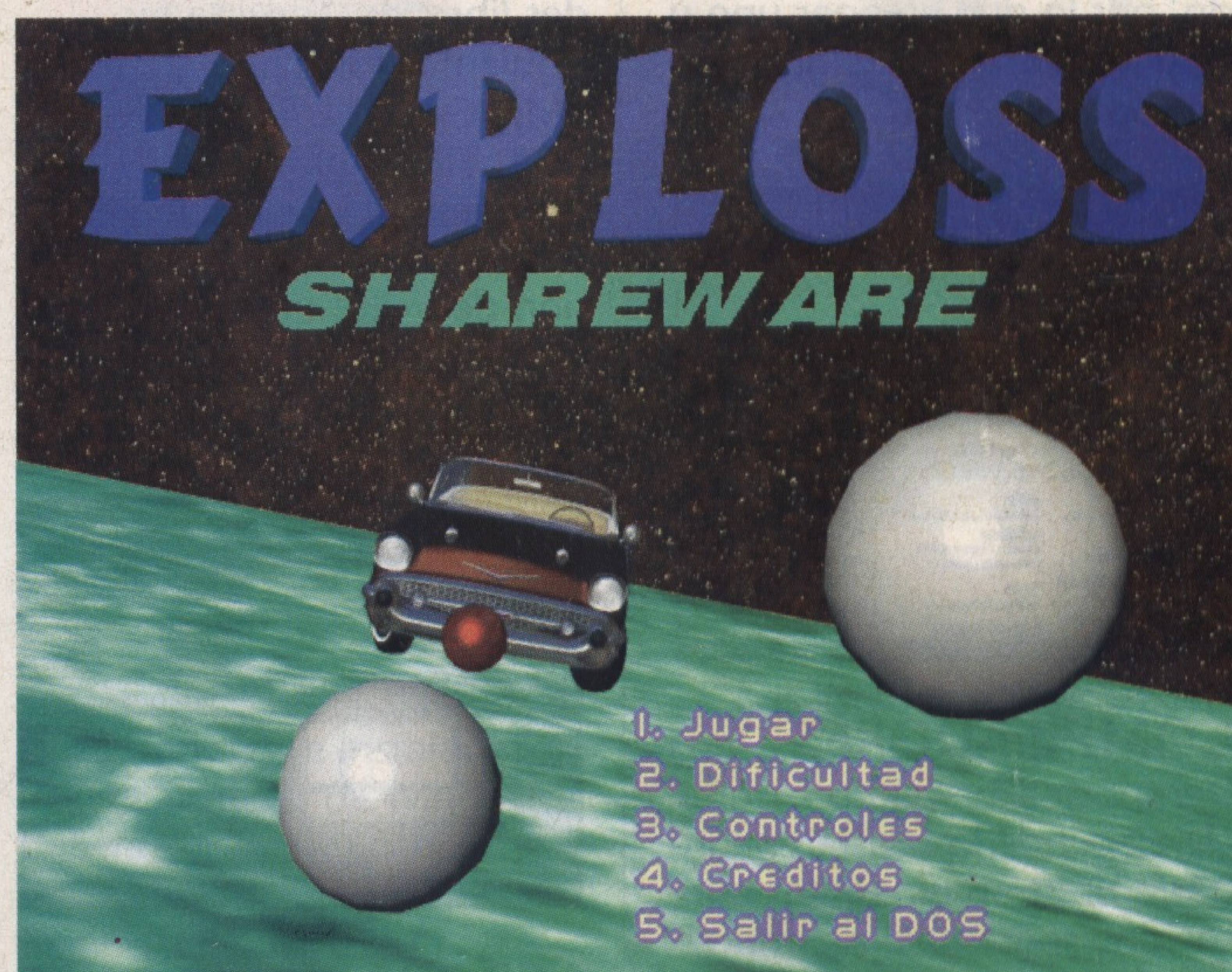
Hemos decido incluir tres librerías muy útiles para quienes deseen internarse en el apasionante mundo de la programación de videojuegos. En la primera de ellas podrás encontrar todo tipo de fuentes, que te servirán para crear tus propios créditos. En la segunda de ellas encontrarás los sonidos más variopintos, aptos para todo tipo de tendencias. En la tercera, por último, dispones de una completa colección de texturas con las que rellenar tus polígonos.

JUEGOS

Dentro de este directorio irán los juegos que se comentan cada mes y han sido premiados. Esperamos todas vuestras aportaciones, a ver si el próximo mes sois vosotros quienes tenéis un juego publicado en esta sección. Pero veamos los ganadores de este número.



QA.



Exploss.

QA

Muchos de vosotros habréis jugado con *Columns*. Este juego es una versión muy buena, realizada en DIV. Espero que os guste, ya que a nosotros nos ha gustado muchísimo; basta con ver que se ha llevado el primer premio.

EXPLOSS

Catalogar este juego es un poco más difícil. Es una mezcla de varios juegos, como *Pintor*, *Pang*, etc. La misión es pintar todo el suelo, evitando unas burbujas asesinas. Al acabar cada fase, se nos premiará con la visión de una bella señorita. Ha sido el segundo premiado.

CINQUILLO

El último programa premiado es de cartas. Todos habréis jugado al conocido juego cinquillo, donde a partir del 5 se deben ir poniendo cartas hasta completar la baraja. La inteligencia del juego es bastante aceptable. Os retamos a todos a tener más de un cincuenta por ciento de victorias en videojuego que no es tan fácil como parece.

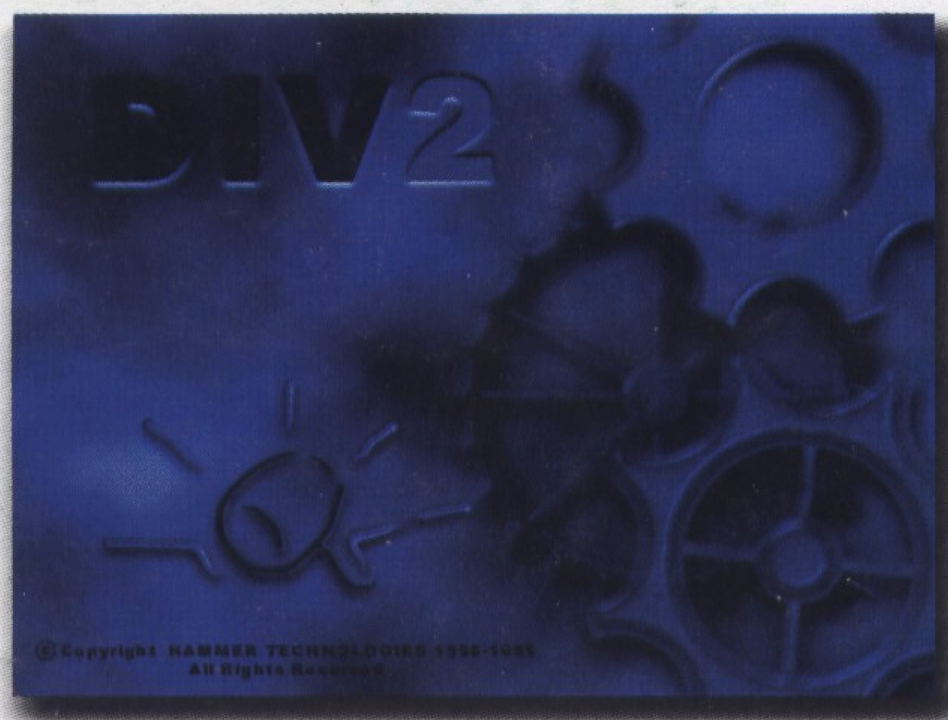


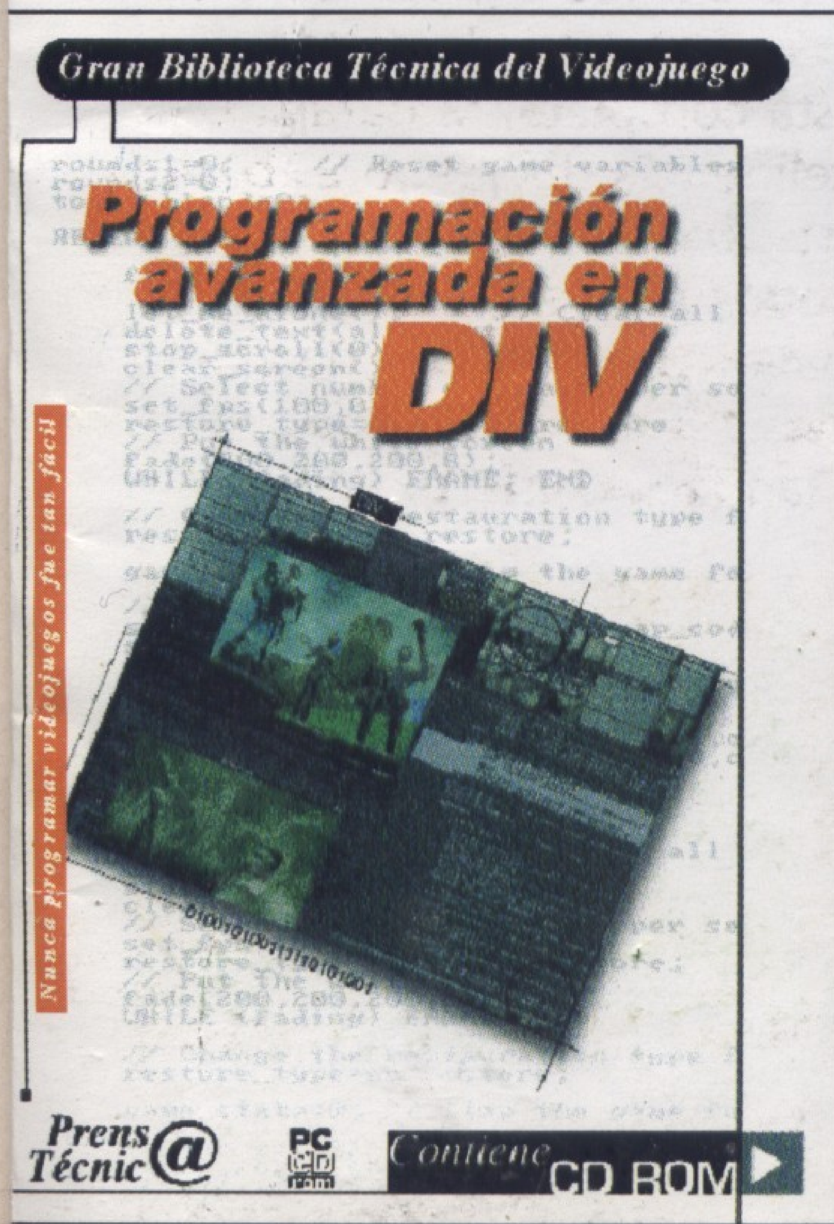
Cinquillo.

10 RAZONES para SUSCRIBIRSE a DIVmanía

Si no quieres quedarte atrás en el mundo de la programación, no lo dudes y suscríbete a DIVMANÍA. Te damos diez buenas razones:

1. **Porque** es la única revista escrita por y para los programadores de videojuegos. Nuestra redacción está compuesta por veteranos desarrolladores, expertos del entorno DIV, grafistas y muchos otros profesionales del software de entretenimiento que dan lo mejor de sí a los lectores.
2. **Porque** es la única revista que te ofrece lo último en el delicado campo de la programación de videojuegos, con los títulos que se encuentran en proceso y los productos *recién salidos del horno* o de los PCs.
3. **Porque** para seguir avanzando hay que saber echar la vista atrás y a la vez no olvidarse del futuro. Y DIV Manía empieza un nuevo camino.
4. **Porque** nuestra revista presenta un *look* muy cercano a sus lectores, salido de las inquietudes de todos vosotros.
5. **Porque** en el interior del CD-Rom encontrarás los elementos con los que todos los programadores sueñan.
6. **Porque** nunca nadie te ha ofrecida tanto por tan poco; nunca has tenido tan cerca la oportunidad de estar al día de lo último en programación por el mínimo esfuerzo de acercarte al kiosco o enviar nuestro cupón y recibir la revista en casa.
7. **Porque** sabes que nosotros somos como tú y conocemos, más o menos, qué se esconde dentro de tu cabeza. Y, si no lo conocemos aún, lo aprenderemos gracias a ti.
8. **Porque** te ofrecemos las más diversas sorpresas, las más interesantes ofertas, para que nunca olvides que la programación siempre está viva.
9. **Porque** prometemos llegar hasta ti con fiel puntualidad, así como tú te acercarás a nosotros. Una puntualidad bimestral, por cierto.
10. **Porque**, por último, los primeros 100 suscriptores de la revista serán, de hecho, los 100 primeros suscriptores, lo que para ellos será un honor. Y para nosotros, claro.





Lo que desde luego llegará a todos los suscriptores son las ofertas especiales que os tenemos preparadas. En primer lugar, los indecisos que sólo se atrevan a unirse a nosotros durante un año recibirán el libro ***Programación Avanzada en DIV***, que abre las miras a los usuarios de DIV.

Y decimos indecisos porque el premio gordo se lo llevarán aquellos lectores que se atrevan a cometer la locura de permanecer a nuestro lado durante dos años. Sí, dos años que se llevarán una merecida Palma de Oro a la Fidelidad: **DIV II**, sin más retórica. Sí, no os habéis equivocado al leer. Os ofrecemos la segunda edición de vuestra herramienta preferida. El compilador de videojuegos que te permite ahora avanzadas funciones 3D y nuevas posibilidades para las redes.

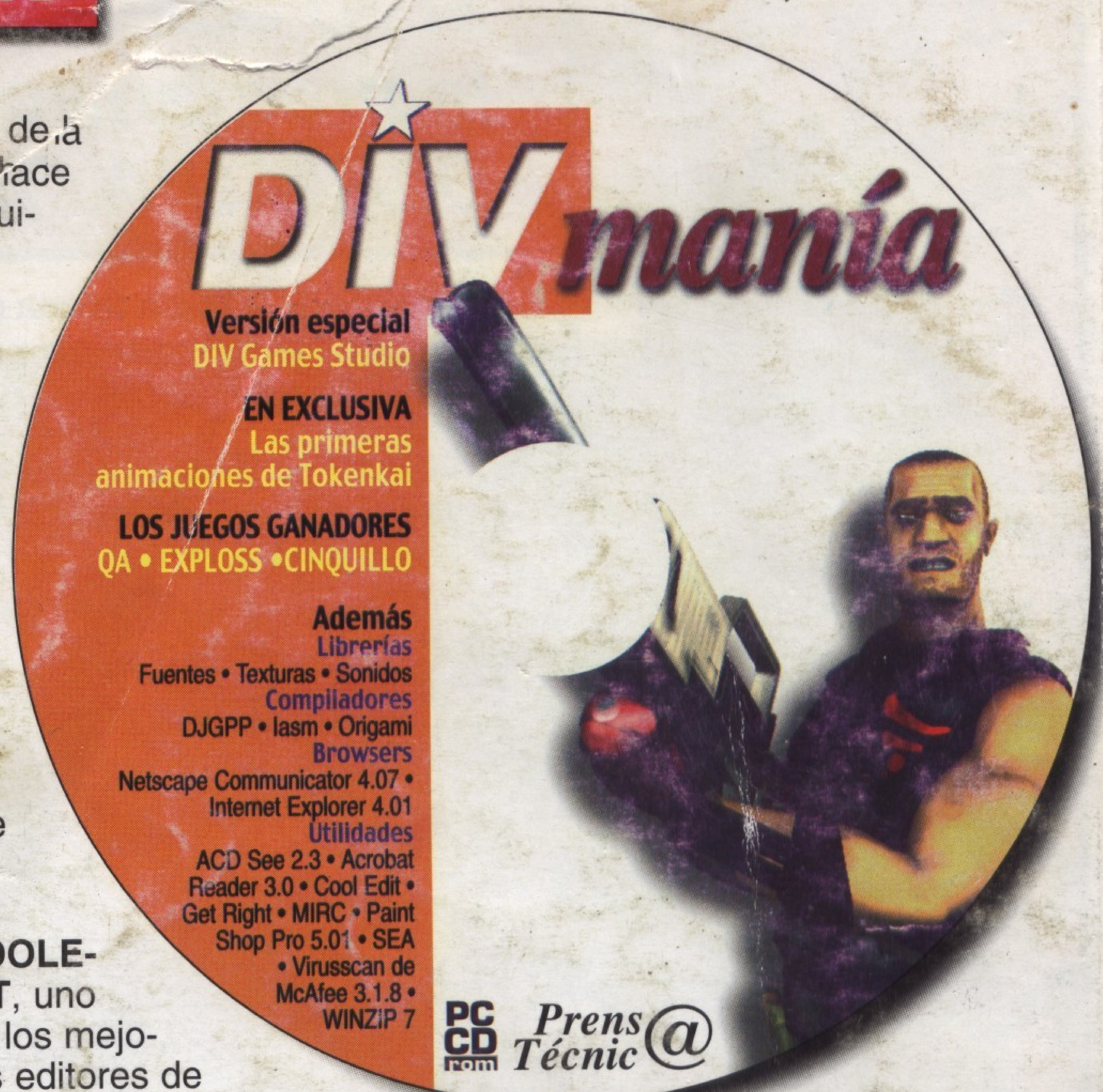


CONTENIDO DEL CD ROM

DEMO DIV GAMES STUDIO
La demo más avanzada de un entorno que ha roto fronteras en el campo de la programación de videojuegos. En esta ocasión la versión que ofrecemos hace especial hincapié en el apartado de los videojuegos, de los que se ha incluido una cantidad considerable.

ANIMACIONES TOKENKAI EN EXCLUSIVA
El primer programa profesional programado con DIV muy pronto se encontrará en el mercado. Se trata de un título de estrategia que encandilará a los amantes de este género. En nuestro CD-Rom de este número incluimos algunos vídeos, en exclusiva, de un videojuego prometedor.

LIBRERÍAS
Incluimos tres librerías útiles para quienes deseen internarse en el mundo de la programación de videojuegos. En la primera de ellas podrás encontrar todo tipo de fuentes, que te servirán para crear tus propios créditos. En la segunda encontrarás los sonidos más variopintos, aptos para todo tipo de tendencias. En la tercera, por último, dispones de una completa colección de texturas con las que rellenar tus polígonos.



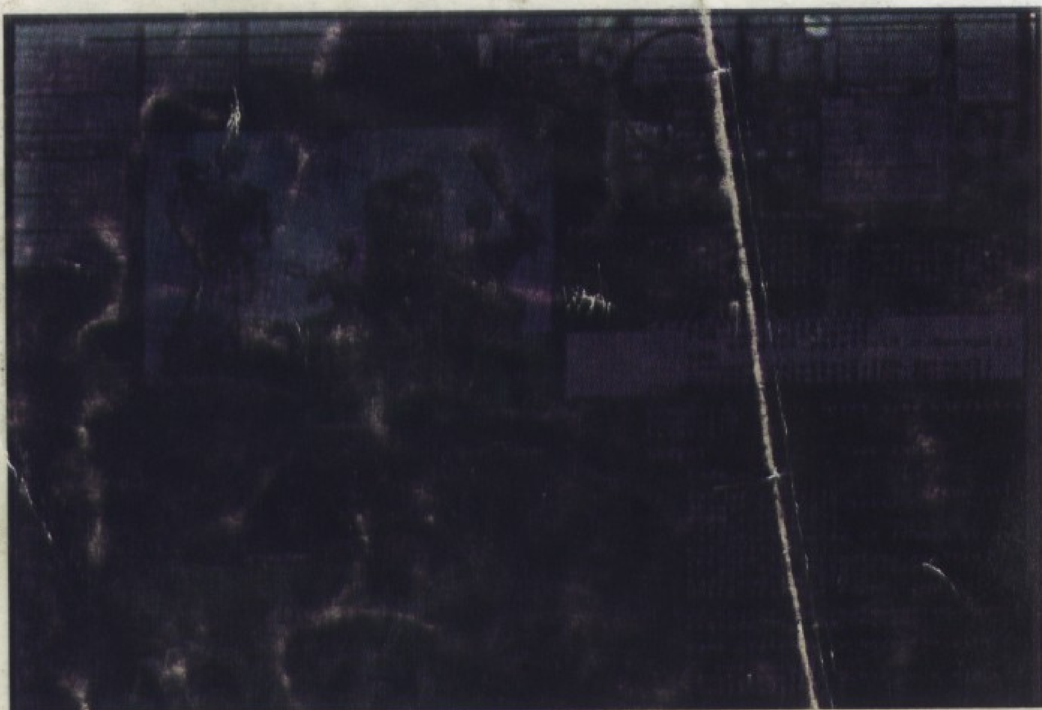
En este primer número podréis dar rienda suelta a vuestras posibilidades como programadores, merced a las utilidades y librerías que os ofrecemos. Los **Browsers** que os ofrecemos son los principales navegadores: Internet Explorer y Netscape Communicator.

Tampoco hemos descuidado el apartado de **Compiladores**, entre los que se encuentran **DJGPP**, un completo sistema de desarrollo de 32 bits en C/C++ para Intel 80386 o superior, **Iasm**, un compilador de ensamblador para plataformas Intel 32 bits y **Origami**, un editor para programadores que soporta cualquier compilador externo (DOS/Windows/Win32) y la utilidad MAKE.

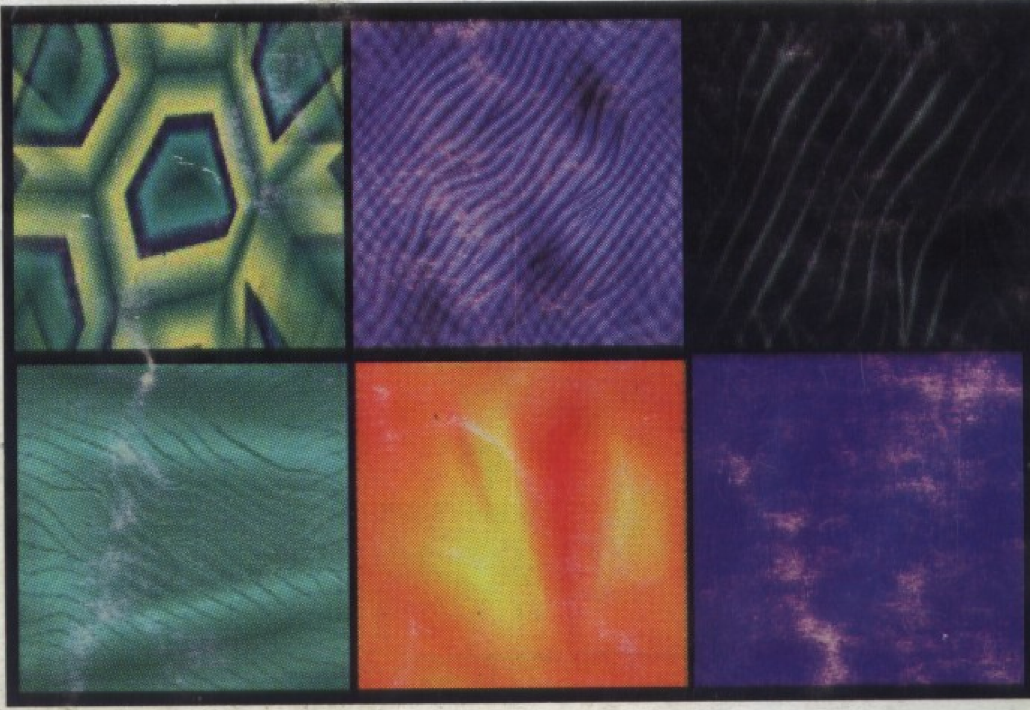
Otras utilidades son: **ACD SEE 2.3**, uno de los visores más conocidos de gráficos. **ACROBAT READER 3.0.**, un lector de ficheros con formato PDF. **Antivirus Anyware para Windows**, que detecta los virus que conoce y los nuevos. **ARJ** en su versión para Windows. **COOLEDIT**, uno de los mejores editores de sonido en su versión shareware. **GET RIGHT**, un programa que sirve para bajar programas de la Red de redes de una forma óptima. **MIRC**, con el que podrás «*chatear*» en Internet. **PAINT SHOP PRO 5.01**, el programa shareware de retoque fotográfico por excelencia. **SEA**, uno de los visores de gráficos dentro del entorno DOS más conocidos. **VIRUSSCAN DE MCAFEE 3.1.8**, un antivirus muy extendido. Y **WINZIP 7**, el compresor de archivos.

Los programas de los lectores que han salido vencedores del concurso son **QA**, un título similar a *Columns*, **EXPLOSS**, mezcla de varios juegos, y **CINQUILLO**, versión del tradicional juego de cartas.

DIV GAMES STUDIO La demo más avanzada de un entorno que ha roto moldes.



LIBRERÍAS Dentro de ellas contemplamos tanto sonidos como fuentes y texturas.



TOKENKAI Animaciones en exclusiva del primer juego profesional realizado con DIV.



DIV manía CON EL MEJOR CONTENIDO


StRAtOS
Un grupo cuya única razón de ser son los videojuegos


TokenKai: estrategia pura
Aparece el primer programa profesional realizado con DIV


El color de la aventura
Cómo entrar en un mundo fascinante

ACTUAL **EXHAUSTIVO** **DIDÁCTICO**

Y MUCHO MÁS...